UNDERSÖKNING AV SIMULERINGSSPRÅKET SIMSCRIPT

Jan Gunnarsson

RE-134 oktober 1973
Inst.för Reglerteknik
Lunds Tekniska Högskola

Undersökning av simuleringsspråket SIMSCRIPT

Examensarbete utfört av

Jan Gunnarsson

Handledare: Björn Wittenmark

Institutionen för reglerteknik
Tekniska Högskolan Lund

Sammanfattning

Examensarbetet avser att undersöka för- och nackdelarna hos simuleringsspråket SIMSCRIPT. Programspråket lärdes med hjälp av manual och läroböcker i SIMSCRIPT (se referenser). Två modeller konstruerades för att undersöka SIMSCRIPT-språkets uppbyggnad. Modellerna beskriver två olika slags system, dels ett, som blott är tänkt och ett, som redan existerar. Den första modellen simulerar arbetet vid ett löpande band. Modellen illustrerar användningen av några för SIMSCRIPT typiska hjälpmedel t ex köer och tidsrutin. Det andra exemplet är en modell av tentamenssystemet vid LTH. Systemet är ganska komplext. Den studiestatistik, som ställs samman, är otillräcklig för att göra en detaljerad modell. Den här använda, förenklade modellen är inte särskilt väl avpassad för indelning av systemet i SIMSCRIPT:s speciella simuleringsvärld och utnyttjar därför inte alla SIMSCRIPT:s fördelar. Hur man använder de specifika SIMSCRIPT-uttrycken framgår däremot tydligt i programmet. Slutligen sammanställs resultat och erfarenheter från SIMSCRIPT-simuleringarna.

Abstract

The master thesis intends to investigate the advantages and disadvantages with the simulation language SIMSCRIPT. The programming language was learnt from manual and books about SIMSCRIPT (see references). Two models were made to examine the SIMSCRIPT language construction. The models describe two different systems, one fictiv and the other already existing. The first model simulates the work at an assembly line. The model illustrates the use of the typical SIMSCRIPT tools as ques and timeroutine. The second example is a model of the test system at LTH. The system is rather complex. The statistics for the studies, that are gathered, are not sufficient to construct a detailed model. The here used, simplified model is not especially well fitted for SIMSCRIPT, and therefore does not use all the advantages of the programming language. The use of the specific SIMSCRIPT statements is however clearly demonstrated in the program. Finally the results and the experiences from the SIMSCRIPT simulations are summed up.

Innehållsförteckning

1. Inledning

Systemsimulering definieras som tekniken att lösa problem genom att följa förändringar med tiden i en dynamisk modell av systemet. Simulering används för att analysera problem, som analytiskt är svåra att lösa. Ekonomiska system, uppbyggnad av datasystem, biologiska och medicinska system och större system, som är svåra och dyra att styra t ex industrianläggningar och rymdprojekt, simuleras.

Man kan vilja simulera av flera anledningar. Nedan följer några exempel:
Inom fysiken beskrivs ofta ett system t ex en komponent i ett värmekraftverk, av ett system av differentialekvationer. Dessa kan vara svåra att lösa och räkningsarbetet kan bli omfattande, då man vill studera hur en parameter påverkar resultatet. Vid medicinska experiment t. ex. då man undersöker, hur ett preparat sprider sig i människokroppen, kan man göra en hypotes och bygga en modell. Man jämför sedan resultaten från modellen med mätresultaten och kan på så sätt förkasta eller verifiera hypotesen. Inom industrin och vid uppbyggnad av datasystem uppstår ofta problem med "flaskhalsar" d. v. s. vissa enheter blir överbelastade och fördröjer arbetet för andra enheter, som får vänta. I en modell av systemet kan man studera hur en ny enhet påverkar "flaskhalsarna" redan innan enheten existerar.

Beroende på systemens natur kan de indelas i olika grupper. Simuleringen kan vara matematisk eller analytisk. Den förra är direkt byggd med utgångspunkt från för systemet gällande ekvationer, medan den senare är konstruerad med hjälp av ett blockschema över systemets komponenter. Om systemets uppförande och konstruktion redan är kända kan modellen göras med systemets olika komponenter som förebild (system design). Är systemets uppbyggnad okänd, måste den provas fram efter olika hypoteser (system prostulation). Beroende på om förändringarna i systemet sker fortlöpande eller vid enstaka tidpunkter indelas de i kontinuerliga och diskreta system.

De olika metoder, som används att simulera system är t ex FORTRAN, speciellt avpassade simuleringsspråk och analogimaskin. För kontinuerliga system används följande metoder:

Analogimaskinen, som utnyttjar analogin mellan matematiska operationer och funktionen hos elektriska komponenter. Den har som bäst en noggrannhet på ca 0,1 %. Hybridmaskinen ger bättre noggrannhet än analogimaskinen, men är långsammare och dyrare.

I The 1130 Continous System Modeling Program (CSMP), byggs problemen upp i blockstruktur.

360/CSMP använder tre olika typer av uttryck:

1) Strukturuttryck, som definierar modellen.
2) Datauttryck, som ger värden till parametrar och konstanter vid programmets början.
3) Kontrolluttryck, som sköter exekvering och bestämmer utskriften.

För diskreta system finns följande programspråk:

SIMULA, som är ett simuleringsspråk byggt på ALGOL.

SIMSCRIPT, som är baserat på FORTRAN.

GPSS (General Purpose Simulation System) är ett språk, uppbyggt av 37 olika blocktyper, som beskriver förlopp i systemet.

Det är svårt att generellt avgöra om FORTRAN eller simuleringsspråken är fördelaktigast att använda, då mycket beror på modellens karaktär och programmerarens skicklighet. Simuleringsspråken är emellertid i de flesta fall att föredra. FORTRAN kräver dock mindre lagringsutrymme och går i regel fortare att exekvera än simuleringsspråken p.g.a. att dessa har en mängd data tillgängliga samtidigt.

Problemlösning med hjälp av simulering kan ske enligt följande mönster:

1. Definition av problemet.

Man bestämmer vilket system, som skall simuleras. Problemet avgränsas, så att ovidkommande delar inte kommer med i modellen.

2. Planering av modellen.

Det material som behövs insamlas. Blockschema över modellen konstrueras.

3. Formulering av matematisk modell.
Hypoteser ställs. Ekvationer som approximerar systemet tas fram.
4. Konstruktion av dataprogram.
Indelning i huvudprogram och subroutiner görs.
Flödesdiagram konstrueras.
5. Verifiering av modellen.
Resultaten från modellen jämförs med mätresultaten.
6. Uppläggning av experiment.
7. Exekvering av simuleringen och analys av resultaten.

Examensarbetet avsåg att undersöka simuleringsspråket SIMSCRIPT. Programspråket lärdes med hjälp av manual och läroböcker i SIMSCRIPT (se referenser). Språkets uppbyggnad, typiska uttryck och dess för- och nackdelar studerades. För två olika system konstruerades modeller som tillämpningsexempel. Det mindre exemplet är en modell av arbetet vid ett löpande band en dag. Exemplet illustrerar användandet av köer och generering av händelser slumpmässigt i tiden. Det visar också hur SIMSCRIPT används i praktiken för att få fram värdefulla uppgifter om en tilltänkt ny enhet i t.ex. en fabrik. Det större exemplet beskriver tentamensresultaten efter två års studier vid LTH för elever med ett visst börjår. Modellen motsvarar ett system, som redan är relativt välkänt. Resultat från systemet finns att tillgå. Systemets uppbyggnad är också bekant (system design). Konstruktion och undersökning av modellen kan anses följa det ovan givna mönstret.

2. Simuleringsspråket SIMSCRIPT

SIMSCRIPT har utvecklats ur två tidigare programspråk. Närmaste föregångare är SPS-1 (Simulation Programming System - 1) hos The Rand Corporation. Detta bygger i sin tur på simuleringsspråket GEMS (General Electric Manufacturing Simulator) konstruerat av H. Markowitz hos General Electric Manufacturing Services. SIMSCRIPT kom till då amerikanska flygvapnet i ett projekt sökte utveckla effektiva simuleringsmetoder. SIMSCRIPT beskrivs på följande sätt i Honeywell Bull Demonstration Guide:

> The key to effective simulation modeling is the SIMSCRIPT system, a powerful, userproven, high-level language, which is in wide usage today. It is an eventoriented algebraic simulation language of FORTRAN - like words, phrases and symbols. It is very generalized to be applicable to wide range of problems to offer great flexibility.
>
> SIMSCRIPT can aid more sound management and decision making by providing more realistic information for evaluation. It can predict the behavior of a physical or abstract process and measure the impact on it of many alternative conditions at a fraction of the cost of implementing the actual process. It translates complex matematical and logical models into meaningful simulation sequences, allowing the user to study the model's behavior under many different conditions.

I SIMSCRIPT tänkes system uppbyggda i en miniatyrvärld, där de ingående komponenterna har motsvarighet i den hypotetiska världen. Det är vid simuleringen fördelaktigt att beskriva systemet med ett antal objekt (entities) med vissa egenskaper (attributes). SIMSCRIPT-världen tänkes konstruerad enligt följande schema:

- STATUS
  - ENTITIES
    - TEMPORARY
    - PERMANENT
  - ATTRIBUTES
  - SETS
- EVENTS
  - EXOGENOUS
  - ENDOGENOUS

Identiteter (entities) representerar föremål t.ex. arbeten, maskiner o s v. Ett system kan ha ett obegränsat antal identiteter av samma och olika slag. Attributen är egenskaper hos identiteterna och antar olika värden. Ett banklån kan t.ex. ha ett attribut för beloppet och ett för förfallodag. Set är köer som relaterar identiteter av samma slag till varandra. Flygplan, som väntar på att få landa på en flygplats kan t.ex. ordnas i ett set, som tillhör flygplatsen. En identitet kan tillhöra ett obegränsat antal set och kan i sin tur även äga ett obegränsat antal set. SIMSCRIPT fordrar att systemet, som skall simuleras, beskrivs uttryckt i identiteter, attribut och set. I programmets definitioner görs två listor över identiteter, en för temporära och den andra för permanenta. Deras attribut anges också och de set, som identiteterna tänkes tillhöra eller är ägda av. De permanenta identiteterna består av räknare och variabler, som har hand om in- och utmatning. Lokala variabler definieras ej. De har värdet noll, då de användes första gången. Temporära identiteter är föremål, som tänkes passera genom systemet och sedan försvinner t ex en bit på ett löpande band. De nedan beskrivna endogena händelserna (events) är också temporära identiteter. Dessa genereras i systemet med anropet CREATE och förstörs med DESTROY.

Events (händelserna) delar in SIMSCRIPT-programmen i subroutiner. En event definieras som det tillfälle i tiden då systemet ändrar status. Händelserna antas ske momentant d.v.s. de tar ingen tid t ex uppkoppling av ett telefonsamtal. Normalt kan en aktivitet representeras med två händelser nämligen början och slut. Händelserna tänkes endast ske då vissa förutsättningar är uppfyllda t.ex. en buss stannar endast då det står passagerare vid hållplatsen och om bussen inte redan är full.

SIMSCRIPT har en tidrutin, som håller reda på när händelserna skall ske. Subroutinen exekveras och tidsrutinen övertar därefter kontrollen. SIMSCRIPT-kommandona CAUSE och CANCEL programmerar resp, inställer händelser i tidsrutinens händelseschema.

Ett SIMSCRIPT-system kan jämföras med ett reelltidssystem. De exogena händelserna motsvaras då av den information reelltidssystemet får, då man kommunicerar med det, och de endogena händelserna av användarens inmatade program.

SIMSCRIPT ger programmeraren möjlighet att specificera de operationer, som måste ske i eventrutiner, såsom ändring av status, start av framtida händelse, exekvering av beslutsregler, lagring och summering av information för systemet samt utskrift av information. Eftersom status består av identiteter, attribut och set, kan den bara ändras, om en identitet genereras eller förstörs, ett nytt värde läses in eller beräknas från ett attribut, eller någon identitet inlemmas eller lämnar ett set. Detta åstadkommes med kommandona CREATE, DESTROY, FILE och REMOVE. Nedan följer en lista över några av de väsentligaste SIMSCRIPT-kommandona:

CREATE temporär identitet eller "event notice" CALLED variabel
DESTROY temporär identitet eller "event notice" CALLED variabel
CAUSE "event notice" CALLED variabel AT decimalt tidsuttryck
FILE variabel IN set
REMOVE FIRST variabel FROM set
REMOVE variabel FROM set
LET variabel = uttryck. Kontrollsatser skilda av komman
STORE uttryck IN variabel. Kontrollsatser skilda av komman
FOR lokal variabel = (uttryck) (uttryck) (uttryck)
FOR EACH
FOR ALL "permanent identitet" lokal variabel
FOR EVERY

IF set IS / IS NOT EMPTY godtyckligt uttryck

FIND variabel MAX / MIN OF utryck

EXOGENOUS / ENDOGENOUS EVENT händelsens namn

REPORT utskriftens namn

En kortbunt för ett SIMSCRIPT-program sammanställs enligt följande:

Run-kort
Styrkort (se appendix 3)
Deklaration av händelsernas typ
(exogena och endogena)
Händelse-rutiner, subroutiner och utskriftrutiner
Styrkort
Systemspecifikationskort

Initialiseringskort

Blankt kort

Kort för start av exogena händelser

End

Definitioner

Definitionskorten beskriver systemets temporära identiteter, deras attribut, vilket set de tillhör samt permanenta systemvariabler d. v. s. räknare och fält definierade för hela systemet. De temporära identiteterna kan vara av två slag nämligen "event notice", markerad med N, och den andra temporära identiteten markerad med T. En "event notice" anger att systemet har en endogen händelse (subroutin) med samma namn. Händelsen kan då läggas in i programmets tidsschema vid olika tidpunkter bestämda av tidigare händelser i systemet.
Den andra temporära identiteten motsvarar ett föremål, som existerar en viss tid i systemet och sedan försvinner.
Dimensionen på deklarerade fält och om variablerna är heltal eller flytande anges efter deras namn. Set kan vara av olika typer nämligen: först in - först ut, sist in - först ut och set, där identiteternas attribut bestämmer turordningen i kön. Beroende på typ av set definieras ett antal variabler, som används för att konstruera det.

Deklaration av händelsernas typ

De exogena händelserna deklareras i en lista med olika namn för varje händelse och händelserna numrerade i löpande följd. Numren används sedan för att starta händelserna (se nedan). De endogena händelserna anges i en annan lista utan numrering. Antal händelser i varje lista skrivs framför listhuvudena EXOG och ENDOG.

Händelserutiner, subroutiner och utskriftrutiner

Varje händelse, som deklarerats finns med som en subroutin i programmet. Vanliga subroutiner av FORTRAN-typ kan också förekomma. Minst en yttre händelse fordras för att starta ett program. Den yttre händelsen kan sedan i sin tur starta inre händelser, då vissa villkor är uppfyllda. Den inre händelsen kan i sin tur starta nya händelser o. s. v. Anrop sker med CREATE (namn), som genererar identiteten och CAUSE (namn) AT (tidpunkt), som anger när händelsen skall ske. Den tid, som förflutit i systemet, finns tillgänglig i variabeln TIME. FORTRAN-subroutiner

kan anropas av händelserna med CALL (namn). Utskrift erhålls genom anrop av en särskild subroutin, REPORT (namn). I denna finns ett antal kort, som bestämmer utskriftens form. Första halvan av kortbunten utgörs av vänsterhandskort, som vart och ett på motsvarande plats i andra halvan har ett högerhandskort. Kort, som innehåller text, som skall skrivas ut markeras med ett X i kolonn ett på vänsterhandskortet. Kortet motsvarar då tillsammans med sitt högerhandskort en rad i utskriften. Om man önskar att värdet av en variabel skall skrivas ut, stansas asterisker, som motsvarar utskriftens längd och placering. På efterföljande kort stansas ett X i kolonn två och variabelns namn skrivs under asteriskerna. Text kan skrivas från och med position sju på vänsterhandskortet till och med position 65 på högerhandskortet. De sista positionerna på högerhandskortet används att styra radframmatning och sidbyte. Position tre till fem används att styra rad- och kolonnrepetitioner.

Systemspecifikationskort

På systemspecifikationskortet anges b1 a om man vill ha initialiseringskorten utskrivna, antal permanenta identiteter och vilket slumptal, som slumptalsgeneratorn skall börja på.

Initialiseringskort

De permanenta variablerna sätts med hjälp av de nummer, som de tilldelats på definitionskorten. De nummer, som svarar mot ett fält, måste även beskriva fältets storlek. Data kan även läsas in från yttre händelse med READ-satser.

Kort för start av yttre händelser

På kortet anges händelsens nummer och när den skall ske. Genom att använda flera kort med olika tidsangivelser kan händelserna anropas flera gånger. Position 13 till 72 eller efterföljande kort kan innehålla data, som läses till programmet.

Nedan följer några exempel för att illustrera det programspråk SIMSCRIPT använder.

En identitet genereras (CREATE IPART) och förstörs (DESTROY IPART).

```
CREATE ARRV
CAUSE ARRV AT TIME +5.0
```

En endogen händelse och en "event notice" har namnet ARRV. Denna subroutin skall startas då TIME har ökats med 5 tidsenheter.

```
LET IQUAL (IPART) = RAND (1,10)
```

Detta tilldelar IPART:s attribut IQUAL slumpmässigt ett heltal mellan 1 och 10.

```
STORE IPART IN IPRT (TEST)
```

Detta åstadkommer att IPART lagras i TEST:s attribut IPRT. Om TEST antas vara en endogen händelse och IPART temporärt förekommande, fordras denna sats för att överföra IPART:s värde till händelsen TEST, eftersom IPART bara är lokalt definierad. I LET-kommandot bestämmer den variabel, som tilldelas nytt värde "mode", medan i STORE-kommandot den variabel, som överför värdet, bestämmer slutligt "mode". STORE-kommandot används alltid vid operationer av den senare typen, för att förhindra att temporära identiteter, som är heltal, lagras som flytande tal.

```
FILE IPART IN QUE
REMOVE FIRST IPART FROM QUE
```

Detta gör att IPART läggs undan i resp tas ur kön.

```
IF QUE IS EMPTY, GO TO 2
```

Detta undersöker om kön är tom.

Do-satsen skrivs på följande sätt

```
DO, FOR K = (1)(MAX)
```

Ett set kan genomsökas på max- och minvärde hos ett attribut eller första identitet med en viss egenskap.

FIND variabel MAX / MIN uttryck

FIND FIRST kontrollsatser

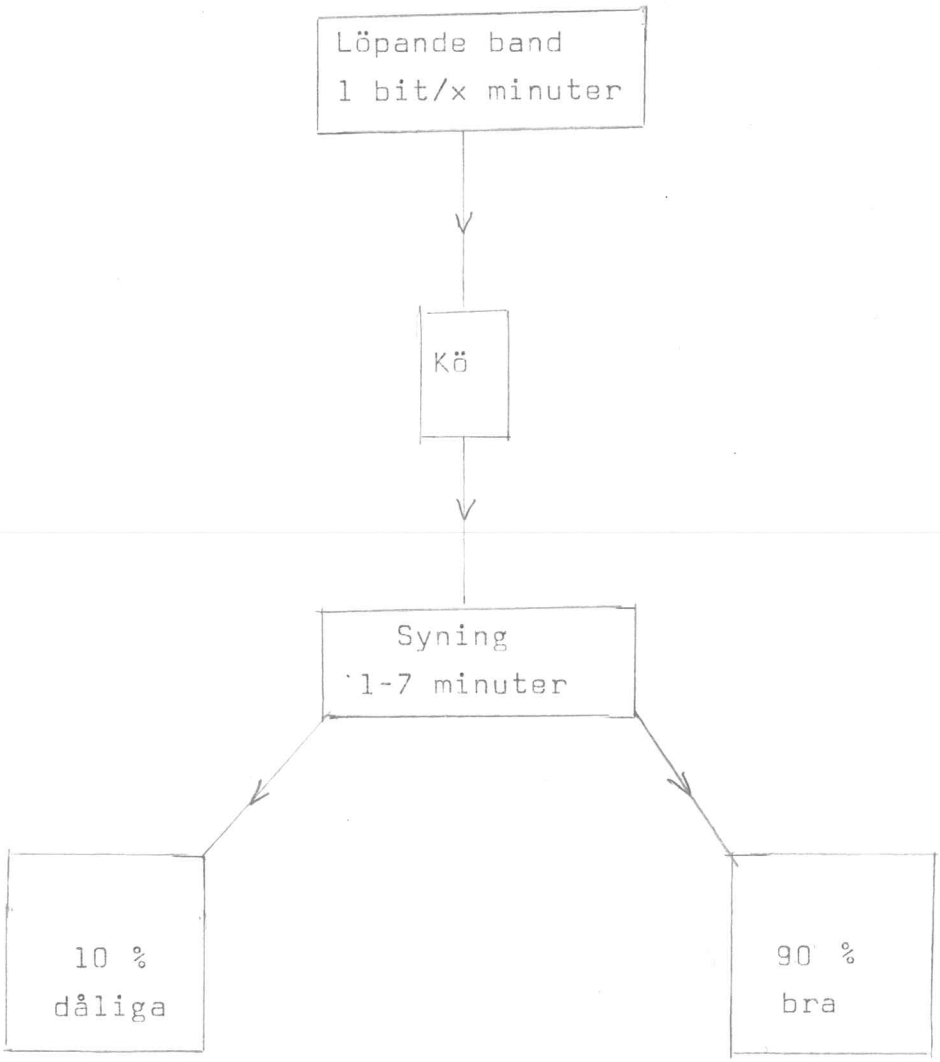
Medelvärde, standardavvikelse och varians kan också enkelt beräknas.

COMPUTE MX, SX, VX = MEAN, STD-DEV,
VARIANCE OF X(I), FOR EACH BASE I

3. Enkelt tillämpningsexempel

Föremålen på ett löpande band kommer med ett visst antal minuters mellanrum. Föremålen synas i en till sju minuter och tio procent av dem underkännes. En arbetsdag med sortering simuleras. Man vill undersöka hur bandhastigheten påverkar bl a väntetider, köer och antal sorterade bitar.

Blockschema över systemet



Detta exempel har praktisk anknytning och skulle kunna tänkas uppstå vid en industri. Man kan då redan innan systemet existerar studera det i en modell. Trots att modellen är enkel kan man erhålla relativt mycket information om systemet till en liten kostnad.

Ett föremål (IPART) definieras som temporär identitet. Det har tre stycken attribut, kvaliten IQUAL, ankomsttiden (ARRVT) och efterföljare i kön (SQUE), som används för uppbyggnad av setet. Två "event notice" definieras nämligen ARRV och TEST. Den förra beskriver ett föremåls ankomst och den senare hur det kontrolleras. Övriga definitioner är kön, QUE, FQUE (först i kön), LQUE (sist i kön), räknare för in- och utmatning och ett attribut till TEST för överföring av värde mellan subroutinerna. Modellen startas av en yttre händelse, START, och avbryts av en annan, STOP, då åtta timmar har gått. START får i sin tur trigga den inre händelsen ARRV.

I subroutinen ARRV görs följande:

IPART genereras. Kvalité och ankomsttid bestäms. Subroutinen undersöker värdet hos en räknare (FREE), som anger om arbetaren är upptagen. Är detta fallet läggs föremålet undan i kön. Står däremot arbetaren och väntar ändras FREE:s värde och TEST sätts att starta efter ett visst antal minuter. Detta antal (en till sju minuter) är den tid det tar att kolla föremålet. Själva subroutinen tar ingen tid att utföra i simuleringsmodellen. Slutligen begär ARRV start av sig själv om MARRV minuter, där MARRV motsvarar det tidsintervall föremål kommer med på det löpande bandet. De båda inre händelserna gör dessutom de beräkningar som fordras för nedan beskrivna utskrift.

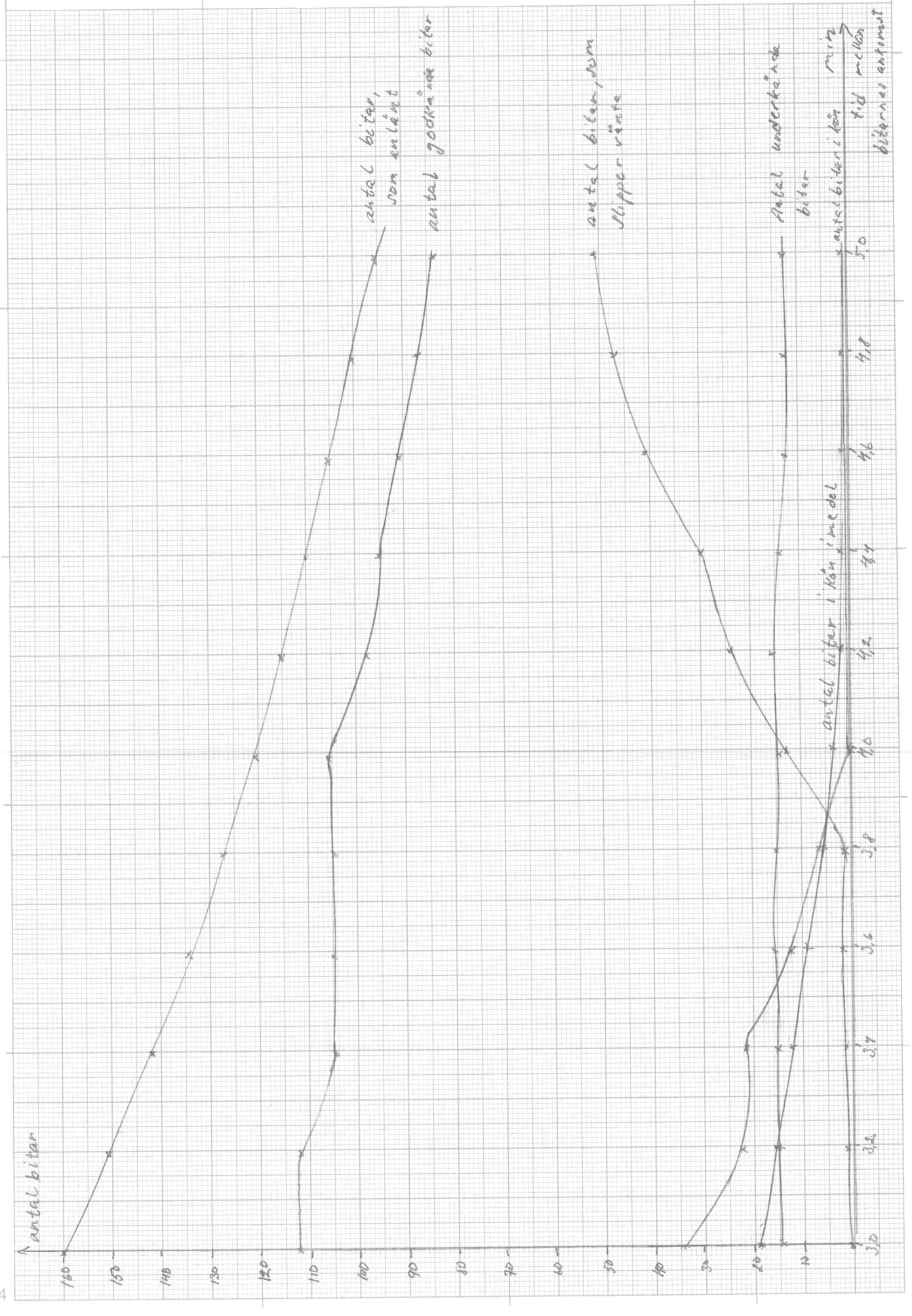
Utskriften består av följande:

Antal minuter mellan föremålens ankomst, antal föremål, antal godkända, antal underkända, hur många föremål som väntar i kön, väntetiden per undersökt föremål i medel, den tid som arbetaren väntat, antal föremål i kön i medel, antal föremål, som inte behövt vänta och arbetsdagens längd i minuter. (Utskrift från programmet, se appendix 1).

Resultat av simuleringen presenteras nedan i två diagram.

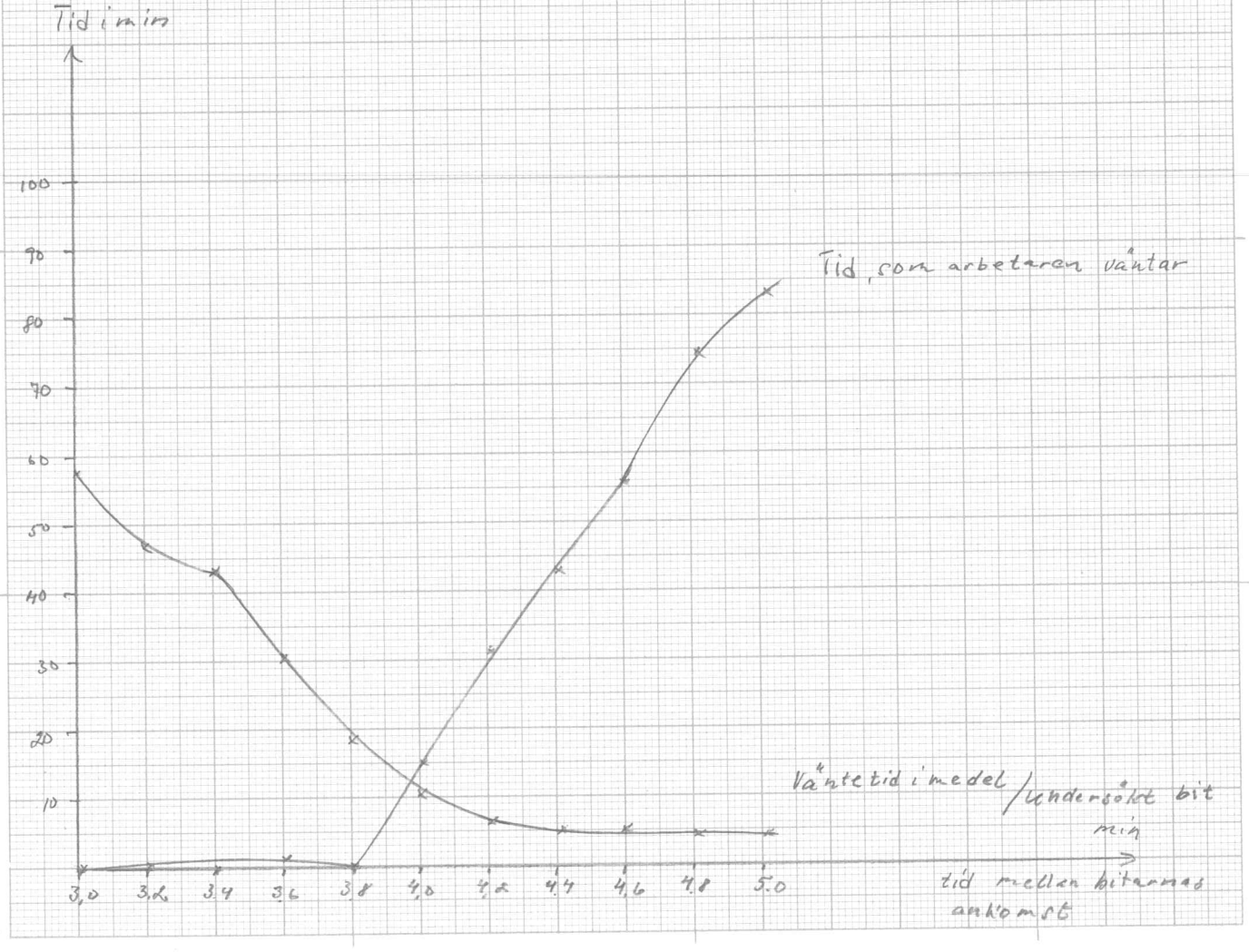
Enkelt tillämpningsexempel

Sorteringsarbete 8 timmarsarbetsdag



Enkelt tillämpningsexempel

Sorteringsarbete 8 timmarsarbetsdag



Av diagrammen framgår att vill man göra arbetarens väntetid liten bör föremålen ej komma saktare än med 3.8 minuters mellanrum. Ökas hastigheten ytterligare till ner mot 3 minuters intervall blir emellertid kön av föremål, som inte undersökts stor. Föremålen får då också ligga längre i kön. Är det viktigt att föremålen slipper vänta eller arbetaren inte har möjlighet att lägga upp någon kö, bör de komma med ca 5 minuters intervall. Men väntetiden för arbetaren ökas då betydligt och färre antal föremål hinner undersökas. Beroende på vilka krav man ställer på sitt system, kan man välja den hastighet på bandet som bäst uppfyller dem.

4. Större tillämpningsexempel

Exemplet simulerar studierna vid LTH för de två första årskurserna på E- och F-sektionen, för elever som börjat höstterminen 1970. Eleverna tenterar i tio olika tentamensperioder. Problemet består i att med hjälp av egen och utgiven studiestatistik som indata, simulera tentamenssystemet och få fram en studiestatistik, som överensstämmer med den som man matade in. Systemet bör naturligtvis även i möjligaste mån, motsvara de erfarenheter man själv har av det.

Systemets uppbyggnad och hur det opererar är välkänt. En del resultat från det finns utgivna i studiestatistik för LTH. Eleverna undervisas i ett antal ämnen i varje läsperiod och tänkes därefter tentera dessa vid periodens slut. En mängd faktorer avgör emellertid om eleven skall tentera. Vilka ämnen har han läst? Hur många nya tentor och hur många gamla kan tenteras i perioden? Passar tentamenstiderna ihop? Tillhör han de bra eller dåliga eleverna? Alla dessa av varandra beroende sannolikheter är svåra att beräkna, men man måste i en modell ta hänsyn till dem för varje elev, betyg, ämne och period. En del av dem kan man anta finns avspeglade i de resultat kansliets betygslistor visar. Antal underkända och godkända för varje period och ämne anges på betygslistorna. De underkända och den återstående delen, de som inte tenterat, utgör, om plussningarna försummas i statistiken, det antal elever, som tenterar i nästa period. Räknar man om fördelningen av antal underkända, godkända och icke-tentare i procent av det antal elever, som kan tentera, får man intervall med olika längder, som motsvarar sannolikheten att en elev erhåller ett visst betyg i ämnet i ifrågavarande period. Om alla elever hade samma sannolikhet att klara sig vore det naturligt att generera slumptal i modellen som fick bestämma elevens betyg. Föll slumptalet inom det intervall, som ett visst betyg upptar i fördelningen erhölls detta betyg. Elevens "betyg" antas även kunna hamna i intervall, som gör honom till icke-tentare eller markerar att han blivit underkänd. Slumpen avgör då elevens handlingssätt för varje period och ämne. Detta motsvarar naturligtvis inte verkligheten. Elevens tidigare tentamensresultat bestämmer i hög grad hans senare studieuppläggning. Vissa elever klarar sig genomgående bättre än medeltalet och andra genomgående sämre. Man kan därför indela eleverna i några grupper

beroende på deras duglighet. Man får då införa någon variabel, som påverkar den inmatade betygsfördelningen, eftersom någon studiestatistik med gruppindelning är svår att åstadkomma. Då man inte vet mängdens sammansättning av de olika grupperna, blir man tvungen att avväga den så att resultatet från modellen stämmer tillfredsställande med verkligheten. Plussningarna kan genereras i modellen genom att låta en viss procent av eleverna inom de olika grupperna tentera, trots att de redan är godkända. De dåliga och bra eleverna antas plussa lika mycket. De bra anses belåtna med sitt betyg och de dåliga hinner ej med att plussa. De medelgoda förmodas därför plussa mest. Hur många, som skall plussa, hur många som skall lyckas, vilka ämnen och betyg det skall beröra, saknas tyvärr statistik för. Det ovan gjorda antagandet är dock rimligt med tanke på att de ämnen, som förekommer i många tentamensperioder och med ett stort antal godkända plussas mest. Plussningarnas antal får man med hjälp av resultatet från modellen korrigera tills de uppskattningsvis överensstämmer med verkligheten. Detta ger en grovt förenklad modell av tentamenssystemet.

Insamling av studiestatistik

Programmet avsåg att simulera de två första årskurserna på E- och F-sektionen för elever, som började höstterminen 1970. Då E-sektionen endast hade statistik över antal godkända i varje ämne och över hur många, som klarat ett visst antal tentor och F-sektionen bara över hur många, som klarat ett visst antal tentor, måste jag själv göra en detaljerad studiestatistik. Denna gjordes på följande sätt:

För E-sektionen

Hundra stycken namn togs ur elevkatalogen för vårterminen 1972. I möjligaste mån undveks namnen på de utlänningar, som jag av egen erfarenhet visste hade börjat i årskurs tre och skrivits in hösten 1970. Frånsett att dessa namn hoppades över, togs de hundra första namnen i alfabetisk ordning. Med hjälp av de betygslistor, som finns på kansliet, gjordes därefter statistiken. Betygslistorna är för varje ämne och läsår arkiverade med ämnena i alfabetisk ordning. Namnen på de godkända eleverna samt elevens

betyg finns angivet på listan. Underkända elever antecknas på baksidan av listan. Om eleven är plussare och plussningen lyckats är ordet "höjning" skrivet vid betyget. I några ämnen är eleverna uppdelade efter börjår. Betygslistorna gicks igenom för de två första årskurserna. Antal underkända, hur många, som blivit godkända med ett visst betyg, samt i vilken period tentamen förekommit antecknades. Med hjälp av namnlistan avgjordes, om eleven börjat 1970. Antal godkända summerades för varje ämne. Därefter kollades om detta antal stämde med det antal godkända, som den utgivna studiestatistiken kommit fram till. Om så ej var fallet, korrigerades mina tal till överensstämmelse. Det antal godkända, som måste läggas till, fördelades så att medelvärdet av de godkändas betyg i varje period förblev konstant. Tentamensperioder med ytterst få eller många godkända korrigerades oftast p g a listor med få namn från den aktuella årskursen lätt gicks förbi och de med många namn lätt räknades samman fel. Eftersom plussningarna inte räknats med, kan bara de elever, som blivit underkända i en period, tentera i den efterföljande. För varje ämne och period räknades därefter antal underkända, godkända och icke-tentare ut i procent av summan av icke-tentare och underkända från föregående period. För varje ämne och period stansades ett kort, där en nolla i position ett anger att ämnet inte kan tenteras i denna period och en etta att det kan tenteras. Position två anger antal underkända, position tre anger antal godkända med betyg tre, position fyra antal godkända med betyg fyra o s v. Position åtta anger antal icke-tentare.

För F-sektionen

Femtionio stycken elever var fortfarande inskrivna efter vårterminen 1972 utav de, som började höstterminen 1970. Namnen på dessa elever togs ur studievägledarens förteckning. I övrigt gjordes statistiken analogt med den för E-sektionen.

Program

(Definitioner och flödesdiagram, se appendix 2!)

Programmet är uppbyggt av en exogen händelse (START), som startar modellen, en inre händelse (TPER), som motsvarar en tentamensperiod och fyra subroutiner, FRSK, som förskjuter betygsfördelningen, KRES, som räknar samman resultaten, PROC, som räknar resultaten i procent och NBTG, som ger eleven betyg. Dessutom finns en utskriftsrutin. Det har inte ansetts nödvändigt att definiera någon temporär identitet, som motsvarar eleven. Elevens duglighet (KVAL) finns i en systemvariabel. Hade KVAL gjorts till attribut till en temporär identitet hade den bara varit lokalt definierad och extra satser hade behövts för att överföra KVAL:s värde mellan subroutinerna. Nedan följer en beskrivning av de olika subroutinerna.

Subroutine FRSK.
Denna subroutine räknar med hjälp av den inmatade studiestatistiken (BTGL) ut en bra fördelning (BTGL3) och en dålig fördelning (BTGL1) enligt följande mönster:
Antal godkända och summan av antal underkända och icke-tentare beräknas. Förhållandet mellan de godkända och summan av de underkända och icke-tentarna räknas ut och multipliceras med KVOT för att erhålla flera godkända i BTGL3. KVOT sätts initialt. Den gamla summan av icke-tentare och underkända minskas med summan av underkända och icke-tentare. Skillnaden fördelas procentuellt på de underkända och icke-tentarna i BTGL och dras därefter från de gamla talen. Den fördelas också procentuellt på de godkända och får öka deras antal. Resultaten läggs i BTGL3. För att erhålla en dålig fördelning divideras förhållandet ovan med KVOT istället för att multipliceras. Resultatet läggs i BTGL1. Om antal godkända är lika med noll, beräknas förhållandet mellan antal underkända och antal icke-tentare. Förhållandet förskjuts sedan på samma sätt som ovan.

Subroutine NBTG (IPER)
Subroutinen får genom anropet reda på vilken tentamensperiod det gäller. Den går sedan igenom de olika ämnena i tur och ordning. Om ämnet inte kan tenteras i perioden är första positio-

nen noll i periodens betygsfördelning. Ett slumptal anger elevens betyg. Beroende på vilken kvalité (KVAL) eleven har tilldelats, tenterar han efter olika betygsfördelningar.

KVAL = 1 medför tentamen enligt BTGL1
KVAL = 2 " " " BTGL
KVAL = 3 " " " BTGL3

Talen i fältet, som beskriver det aktuella ämnets betygsfördelning jämförs med det slumptal, som anger elevens betyg, och adderas till deras summa blir större än slumptalet. Då tilldelas det endimensionella fältet KLAR ett betyg för ämnet. Om slumptalet är större än summan av antal underkända och godkända erhålls inget betyg. Eleven har då inte tenterat. Ett slumptal anger om eleven skall plussa, förutsett att han är godkänd tidigare. Om slumptalet, som anger betyget, visar att han är icke-tentare, plussar han inte. Om slumptalet, som anger om eleven skall plussa är större än ett visst tal, plussar han inte. Plussningsslumptalet är en funktion av kvalitén. Kvalité 1 och 3 plussar minst, 2 mest. Plussningarna sker enligt betygsfördelning BTGL3. Antal plussningar och antal lyckade plussningar för varje ämne beräknas. Antal tentamenstillfällen totalt räknas samman. Antal elever av varje kvalité summeras.

Subroutine KRES

Denna subroutin räknar med hjälp av KLAR ut hur många elever som är godkända, underkända, icke-tentare och hur många, som erhållit ett visst betyg. Dessutom räknar den ut hur många elever, som klarat ett visst antal tentor, både innan tentorna viktats och efter viktning. Viktningen innebär att långa kurser multipliceras med två. Antal godkända elever av en viss kvalité summeras för varje ämne.

Subroutine PROC

Denna subroutine räknar antal elever med ett visst resultat i procent av samtliga. Den räknar också ut hur stor procent ett visst antal tentor utgör av samtliga (med och utan viktning). Antal elever (i procent), som klarat ett visst antal tentor (viktat resultat) räknas dessutom ut kumulativt.

EXOG. EVENT START
Den yttre händelsen anropar FRSK. Denna räknar ut den förskjutna betygsfördelningen. Elev nr 1 tilldelas en kvalité. Elevens kvalité noteras. Slutligen startas TPER upp av subroutinen.

ENDOG. EVENT TPER
Den inre händelsen räknar antal perioder för varje elev. Händelsen startar upp sig själv med 2.4 månaders mellanrum. För varje period anropas NBTG(IPER). Om periodantalet är lika med tio, anropas KRES. Därefter ges KVAL ett nytt värde, som noteras i fältet KV_i för att räkna samman antal elever av varje kvalité. Man får då en ny elev. Periodräknaren nollställs. Antal elever som passerat systemet räknas. Om detta antal är lika med antal elever i årskursen anropas PROC och slutligen anropas FINAL.

REPORT FINAL
Denna subroutine ger utskrift.

Nedan följer några kommentarer och förtydliganden till programmet:
E-sektionen har 114 elever och F-sektionen 59 stycken i den simulerade modellen. En elev genereras genom att kvalité (KVAL) tilldelas ett värde. Han går därefter igenom samtliga tentamensperioder och förstörs då KVAL byter värde. Han erhåller betyg genom anrop av NBTG. Hans betyg läggs undan i det endimensionella fältet KLAR. Sammanräkning av betyg sker i subroutinen KRES. Plussningarna uppskattas till en viss procent av samtliga tentamenstillfällen. Antal plussningar räknas samman i subroutinen NBTG. Beroende på vilken kvalité eleverna har får de tentera efter olika betygsfördelningar. Detta medför att de dåliga eleverna kommer att klara få tentor och de bra många. Om systemet simuleras utan någon indelning i grupper med olika möjligheter att klara sig, stämmer fördelningen av antal klarade tentor per ämne och betyg relativt bra med statistiken. Fördelningen av antal klarade tentor per elev ligger emellertid slumpmässigt spridd inom ett litet område kring hälften av det antal ämnen, som ingår i modellen. För att hålla fördelningen av antal godkända per ämne och betyg relativt oförändrad, antaś den simulerade modellen innehålla två lika stora grupper av dåliga och bra elever. Dessa

elever klarar i medeltal lika många tentor som de medelbra eleverna gör. Mängdens sammansättning bestäms av ett tal kallat FRD, som anger de dåliga och bra eleverna i procent av hela mängden. Hur stor skillnaden skall vara mellan de olika kvalitetsgrupperna, bestäms av ett tal kallat KVOT. KVOT reglerar hur kraftigt de olika betygsfördelningarna skall förskjutas. Fördelningen av antal klarade tentor justeras med hjälp av KVOT och FRD tills den acceptabelt överensstämmer med studiestatistikens fördelning. Plussningarna har försummats i studiestatistiken och åstadkommes av systemet. De kommer endast att förskjuta fördelningen av de redan godkända och har liten inverkan. Om eleven skall plussa antas bero på hans kvalité, därför genereras ett slumptal, som funktion av kvalitén. Är slumptalet större än ett visst tal (FP), sker ingen plussning. De medelbra eleverna antas plussa mest. De dåliga eleverna anses ha fullt upp med att bli godkända och de bra erhåller ett bra betyg första gången de tenterar. Storleken på FP, regleras tills plussarna uppskattningsvis är tillräckligt många. Detta kan vara vanskligt att göra eftersom ingen statistik gjorts över plussningarna. De som plussar antas ha större chans att klara sig än övriga. De tenterar därför efter samma fördelning, som de bra eleverna. Simuleringen avser de två första årskurserna d v s tio tentamensperioder. Den exakta tiden för perioderna är ointressant. De antas därför återkomma med ett intervall på 2.4 månader. Antal ämnen är för E-sektionen 17 och för F-sektionen 20 stycken. Laborationskurser och inledande kurs i tillämpad elektroteknik, som nästan samtliga elever klarar vid en tidpunkt, har inte tagits med vid simuleringen.

5. Resultat av simuleringen och jämförelser med verkligheten

(Exempel på programutskrift, se appendix 2!)

Modellens parametrar är KVOT och FRD. Med hjälp av dessa justerades modellen in i ett läge så att antal godkända tentor per ämne och betyg och antal godkända tentor fördelade på eleverna acceptabelt överensstämde med verkligheten. KVOT hade då värdet 4.50 och FRD 33.3. Det i modellen ingående begynnelseslumptalet (se nedan) hade satts till 321453. Jämförelse mellan betygsfördelningen för varje ämne i modellen och enligt statistiken har gjorts i tabell 1 (för E-sektionen) och tabell 2 (för F-sektionen). För E-sektionen finns dessutom studiestatistik utgiven över hur antal klarade tentor fördelar sig på eleverna. Denna jämförs med modellens fördelning i diagram 1. F-sektionen saknar motsvarande statistik, men modellens fördelning av antal klarade tentor på olika elever framgår av diagram 2. KVOT och FRD har samma värde som för E-sektionen. En ökning av KVOT kan kompenseras med en minskning av FRD d v s görs skillnaden mellan eleverna inom de olika grupperna stor, måste de bra och dåliga elevernas antal minskas. Att få god överensstämmelse med många medelbra elever och få bra och dåliga är dock svårt, eftersom KVOT då måste göras mycket stort. Förskjutningen blir då så stor för de bra och dåliga eleverna att flertalet av dem med stor sannolikhet blir god- resp. underkända. Modellen blir då ganska okänslig. Fördelningen av antal klarade tentor på olika elever kommer då slumpmässigt att ligga spridd kring tre approximativt lika stora områden, som knappast överlappar varandra alls. Det är därför bättre att minska KVOT och göra FRD större. Då de bra, dåliga och medelbra eleverna är lika många och KVOT = 4.5 är överlappningen markant. Ökas FRD så att den blir mycket större än 1/3, distorderas fördelningen av antal godkända tentor per ämne.

Tabell 1.

Jämförelse mellan simulerat värde och statistiskt värde (inom parentes) för E-sektionen (räknat i procent).

Ämne	GODK	UDRKD	BTG3	BTG4	BTG5	BTG6	BTG7	EJTNT
LIN.A.	71(76)	26(-)	24(32)	32(38)	12(14)	4(2)	0(0)	3(-)
FSK.I.	74(82)	17(-)	44(49)	22(23)	8(9)	0(1)	0(0)	10(-)
ANL.1	76(77)	18(-)	48(49)	18(17)	10(10)	1(1)	0(0)	6(-)
STAT.	66(74)	30(-)	45(48)	21(23)	0(3)	0(0)	0(0)	4(-)
V.O.AT.	67(76)	18(-)	25(37)	25(25)	15(12)	2(2)	0(0)	16(-)
DYN.1	62(56)	14(-)	33(35)	22(15)	7(6)	0(0)	0(0)	24(-)
ANL.2	59(50)	25(-)	38(32)	14(14)	6(2)	1(2)	0(0)	17(-)
VKTRA	68(69)	4(-)	32(36)	23(20)	13(13)	0(0)	0(0)	29(-)
NUM.A.	61(64)	19(-)	33(36)	23(25)	5(1)	0(2)	0(0)	20(-)
KRFSK	60(54)	8(-)	21(22)	32(27)	7(5)	0(0)	0(0)	33(-)
DYN.2	59(53)	24(-)	37(31)	21(18)	1(4)	0(0)	0(0)	18(-)
KVNTM	42(32)	10(-)	13(14)	14(9)	14(8)	1(1)	0(0)	48(-)
MT.ST.	42(38)	11(-)	20(19)	13(11)	9(8)	0(0)	0(0)	47(-)
TR.EL.2	32(24)	17(-)	26(20)	5(3)	0(1)	0(0)	0(0)	52(-)
TR.EL.1	39(30)	24(-)	27(23)	11(6)	1(1)	0(0)	0(0)	38(-)
SLHT	51(48)	22(-)	26(25)	21(19)	4(4)	0(0)	0(0)	27(-)
FUNK.	51(40)	16(-)	29(23)	14(11)	8(5)	0(0)	0(0)	33(-)

Värden som saknas markeras med ett streck.

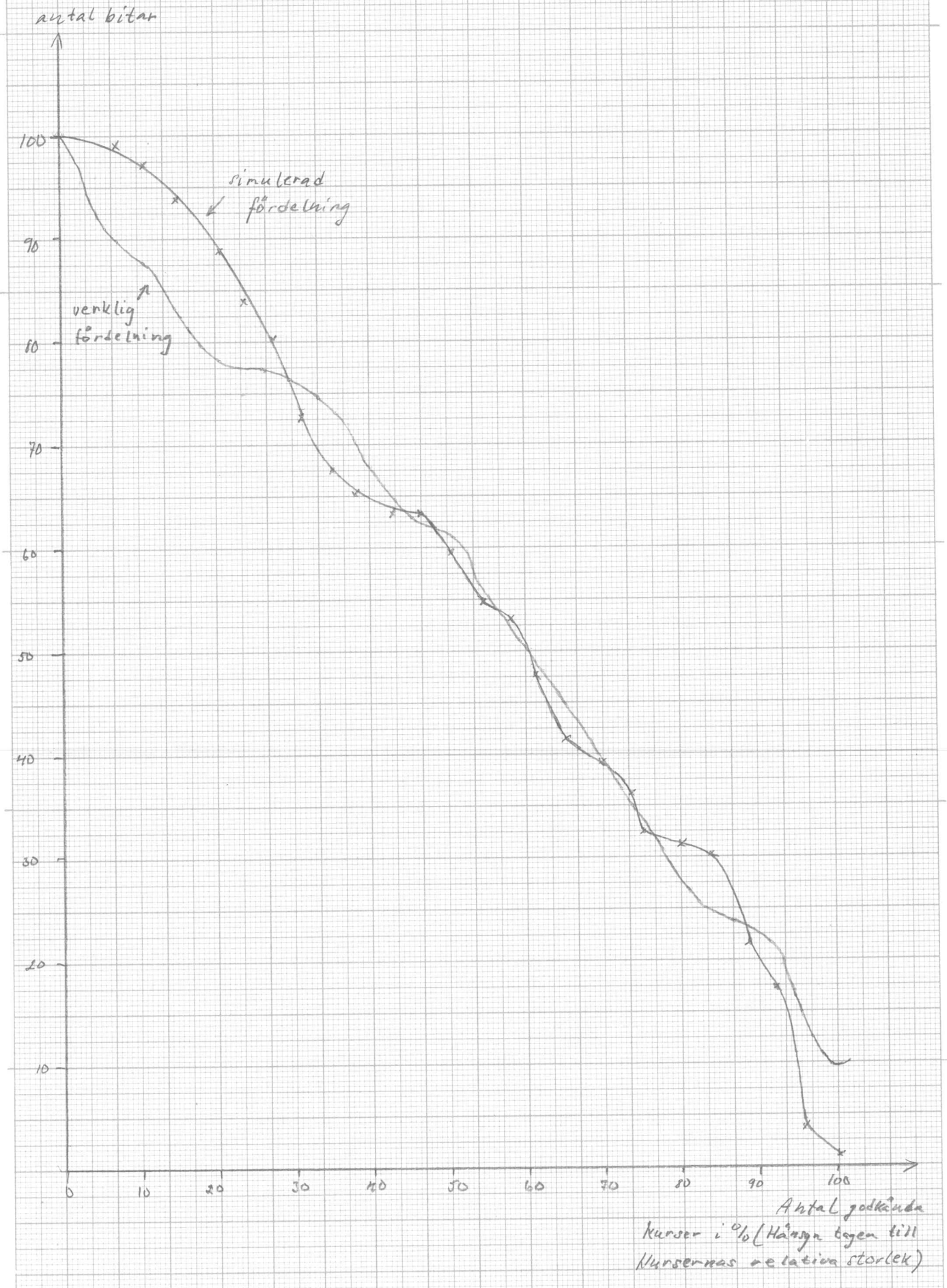
Tabell 2.

Jämförelse mellan simulerat värde och statistiskt värde (inom parentes) för F-sektionen (räknat i procent av samtliga i årskursen).

Ämne	GODK	UDRKD	BTG3	BTG4	BTG5	BTG6	BTG7	EJTNT
FSK.I	75(81)	12(5)	24(32)	42(42)	7(5)	2(2)	0(0)	14(14)
HÅLLF.	51(61)	37(24)	12(20)	25(29)	10(8)	3(3)	0(0)	12(15)
ANL.1	68(86)	27(5)	36(34)	12(29)	20(24)	0(0)	0(0)	5(9)
ANL.2	63(76)	17(7)	27(24)	25(36)	5(10)	5(7)	0(0)	20(17)
LIN.A.	76(92)	20(0)	25(25)	32(44)	17(19)	2(3)	0(0)	3(8)
VKTRA	63(81)	5(2)	17(20)	31(27)	14(15)	2(2)	0(0)	32(17)
STAT.	68(78)	25(8)	44(39)	19(29)	5(10)	0(0)	0(0)	7(14)
VÅGLR.	20(20)	25(14)	17(14)	3(5)	0(2)	0(0)	0(0)	54(66)
DYN.1	61(63)	32(14)	31(25)	15(19)	15(19)	0(0)	0(0)	7(23)
NUM.A.	63(66)	14(7)	22(20)	24(31)	14(14)	3(2)	0(0)	24(27)
RIT.TK.	83(90)	5(2)	31(42)	53(46)	0(2)	0(0)	0(0)	12(8)
MA.FK.	51(51)	3(3)	20(20)	25(17)	5(14)	0(0)	0(0)	46(46)
FUNK.	46(51)	9(14)	15(20)	17(19)	14(12)	0(0)	0(0)	46(35)
KVNTM	53(46)	3(0)	14(12)	20(15)	19(19)	0(0)	0(0)	44(54)
SLHT	64(70)	14(8)	19(22)	22(25)	24(22)	0(0)	0(0)	22(22)
MT.ST.	36(39)	2(5)	10(15)	9(8)	17(15)	0(0)	0(0)	63(54)
DYN.2	66(64)	9(7)	5(12)	41(32)	20(20)	0(0)	0(0)	25(29)
TR.EL.1	37(31)	19(10)	19(17)	14(8)	5(5)	0(0)	0(0)	44(59)
TR.EL.2	34(24)	31(22)	24(17)	5(5)	5(2)	0(0)	0(0)	36(54)
MVTMS	41(39)	0(3)	15(14)	15(8)	10(17)	0(0)	0(0)	59(58)

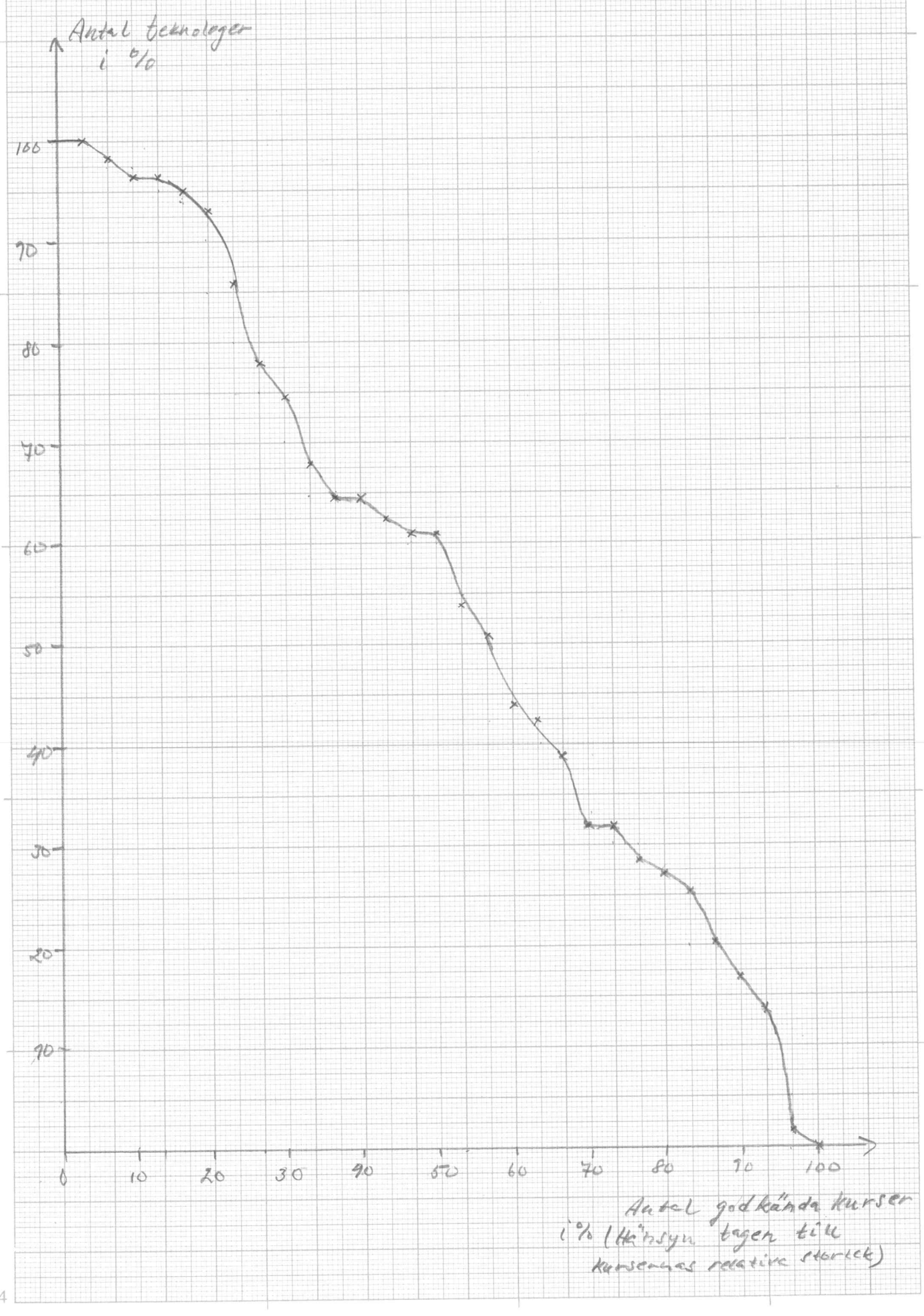
Diagram 1

Kumulativt histogram för teknologer intagna höstterminen 70. E-sektionen.



Kumulativt histogram för teknologer intagna hösterminen 70. F-sektionen.

Diagram 2



Av tabell 1 framgår att överensstämmelsen mellan modell och verklighet är god, om ämnet har tenterats många gånger (tabellens början). Medan de ämnen som varit uppe få gånger (tabellens slut) stämmer sämre. I tabell 2 kan man också se att simuleringen för F-sektionen, som har ca hälften så många elever som E-sektionen, är skillnaden mellan simulerat resultat och verklighet större än i tabell 1. Detta beror på att betygen i modellen genereras av rektangelfördelade slumptal. Ett betyg måste genereras relativt många gånger för att modellens resultat skall avvika lite från statistiken. Tentamensresultaten är bättre för F-sektionen än för E-sektionen, vilket är i överensstämmelse med verkligheten. Fördelningen av antal klarade tentor på olika elever visar att den simulerade kurvan för E- och F-sektionen är ganska lika. E-sektionens kurva stämmer godtagbart med statistikens. F-sektionsfördelningen antas vara likartad eftersom det inte finns några verkliga resultat att jämföra med.

En intressant egenskap hos en modell är dess beroende av parametervariationer. Om man varierar KVOT, som beskriver skillnaden mellan olika elevgrupper, erhåller man en liten ändring. Ändringen påverkar resultatet mindre om KVOT redan är ett stort tal. Den påverkar inte de medelgoda elevernas betyg. Förskjutningen av antal klarade tentor enligt diagram 1 och 2 blir därför begränsade till de bra och dåliga eleverna. Är KVOT redan relativt stort, ligger eleverna inom dessa grupper spridda kring en tydlig topp, som inte förskjuts mycket av en ändring i KVOT.

Modellens andra parameter är FRD. Den beskriver de bra och dåliga elevernas antal i procent av hela mängden. Ändras FRD påverkas resultatet märkbart. Kurvorna i diagram 1 och 2 vrids uppåt i övre delen och nedåt i nedre delen. Ändringen är emellertid inte särskilt stor. Däremot ändras tabell 1 och 2 genomgående. Eleverna tilldelas slumpmässigt sina kvalitéer i modellen. Fördelningen blir den som FRD beskriver, om eleverna är tillräckligt många. Ändringen av FRD visar emellertid att snedfördelningar blir vanliga. En liten ändring av FRD förskjuter därför ofta inte mängdens sammansättning i någon bestämd riktning.

Modellen är som synes beroende av de slumptal, som genereras i den. SIMSCRIPT-språkets slumptalsgenerator behöver som utgångspunkt ett udda, sexsiffrigt begynnelsetal, som sätts av programmeraren eller konstant är lika med 777 777. Varje följande slumptal genereras sedan med närmaste föregående som utgångspunkt. Ändras begynnelsetalet kommer alltså samtliga slumptal i modellen att bytas ut. Detta medför att mängdens sammansättning och betygsfördelning påverkas. Fördelningen av antal godkända tentor per ämne ändras mer än kurvorna i diagram 1 och 2, som i regel bara får mindre spridning av antal klarade tentor d v s de medelgoda elevernas antal ökas. Man kan även låta bli att slumpmässigt indela mängden i grupper. Dessa får istället utgöra exakt det procenttal av mängden, som FRD anger. Betygsfördelningen och fördelningen av antal klarade tentor påverkas då mindre av en ändring i begynnelseslumptalet, men fortfarande erhålls betydande variationer. Resultaten från simuleringarna varierar på ca 10 % kring de statistiska värdena d. v. s. man måste ta medelvärdet av relativt många simuleringar för att få god överensstämmelse med verkligheten.

Plussningarna har i programmet satts till ett visst procenttal av ett slumptal genererat som funktion av kvalitén. Detta tal har i modellen uppskattats till åtta procent. Fördelningen av antal plussningar och antal lyckade plussningar per ämne, se programutskriften i appendix 2!

Resultatet av programmet ger, trots dess känslighet för variationer i begynnelseslumptalet och FRD, med tanke på de grova approximationerna i modellen, en relativt god bild av den inmatade studiestatistiken. Den förändring, som skulle kunna göras i modellen är att framhäva elevens identitet bättre. Man kan indela mängden i flera grupper av elever med olika duglighet och låta eleven byta kvalité om vissa villkor är uppfyllda. Skillnaden mellan olika grupperna blir då mindre och gruppindelningen vid programmets början blir inte så känslig för en snedfördelning. Modellen blir dessutom mer differentierad. Betygsfördelningen kan göras mindre beroende av begynnelseslumptalet genom att generera flera olika slumptal och låta vart och ett bestämma ett tal, som motsvarar ett visst antal poäng på en tentamensskrivning.

Programmet simulerar en följd av händelser, som i verkligheten beror av varandra på ett sådant sätt att det är svårt att göra någon statistik för beroendet. Elevens tidigare resultat bör påverka hans senare handlingssätt, såsom sker i verkligheten. I programmet sker detta genom att vissa elever antas få bättre eller sämre tentamensresultat än andra. Tentamensperioderna kommer emellertid med approximativt konstanta intervall och oberoende av vad som skett i föregående period. Eleverna tenterar oberoende av varandra. Med den uppbyggnad systemet har, kan det med fördel simuleras i FORTRAN. Programmet hade inte blivit enklare att skriva än i SIMSCRIPT, men det hade förmodligen gått fortare att exekvera. I en mera detaljerad modell, där elevernas intagning till LTH tagits med i simuleringen, kan det vara fördelaktigt att lägga upp en kö för eleverna. Programmet kan då utökas med flera subroutiner, som ur kön tar ut de elever, som skall läsa ett visst ämne och de, som sedan skall tentera det. Man kan också låta vissa attribut hos eleven innehålla värden, som får avgöra hans senare beslut inför nya tentamensperioder. En sådan modell skulle avgjort lämpa sig bättre för SIMSCRIPT-simulering än för FORTRAN-simulering. Ett tänkbart syfte att bygga en modell av LTH:s tentamenssystem kunde vara, att man ville studera hur omplaceringen av ämnen mellan olika tentamensperioder påverkar resultaten. En detaljerad modell, där man i simuleringen tagit med några av de vanligaste studiesituationerna och låtit elevens delresultat avgöra vilka ämnen han skall läsa och tentera, skulle kunna ge ett resultat, som är närliggande systemets.

6. Erfarenheter och sammanfattning

SIMSCRIPT är speciellt konstruerat för att man enkelt skall kunna bygga modeller av olika system. Programspråket är byggt på FORTRAN och är lätt att lära om man kan FORTRAN. En del uttryck i FORTRAN bl. a. för in- och utmatning kan blandas med SIMSCRIPT-språket. Utöver SIMSCRIPT:s FORTRAN-grund finns naturligtvis en specifik SIMSCRIPT-del. Nedan följer en beskrivning av några av för- och nackdelarna med att simulera i SIMSCRIPT.

Indelning i identiteter, attribut och set bildar en speciell SIMSCRIPT-värld. Uppläggning av ett system i en modell beror väl i viss mån på programmeraren, men är ofta så klar, att den inte kan göras på mer än ett sätt. Då man löst ett antal problem, ser man att flera av dem, motsvaras av samma typ av modell i SIMSCRIPT. Det enkla tillämpningsexemplet skulle t.ex. med små ändringar lika gärna kunna simulera kunder i kö framför en postlucka eller personer, som köar framför en kaffeautomat.

SIMSCRIPT innehåller en tidsrutin, som håller reda på tiden i modellen. Tidsrutinen har ett eget tidsschema, där händelser sätts att ske vid framtida tidpunkter. Exogena händelser tillåter även att man ingriper i modellen vid en vid programmets början förutbestämd tidpunkt. Tiden finns tillgänglig vid varje tidpunkt i en variabel, TIME, som även kan skrivas ut vid programmets slut. I FORTRAN hade man själv varit tvungen att lägga in en sådan rutin i sitt program.

Köer kan läggas upp i SIMSCRIPT i set. Identiteter som läggs i kön, kan tas ut efter olika mönster. Först-in - först-ut, sist-in - först-ut och set, där en viss egenskap hos identiteten avgör vilka, som skall ha företräde. Köer kan också avsökas med avseende på identiteten med ett visst attribut med hjälp av ett specialuttryck. Medelvärde, standardavvikelse och varians för ett set kan enkelt beräknas med hjälp av en enda sats. Identiteten läggs i kön eller tas ur den med lätt förståeliga kommandon, som består av en enda sats. Temporära identiteter är bara lokalt definierade, men har man fört över identiteten till en ny subroutine så existerar också dessa samtliga attribut. Identiteter som ligger i kö är åtkomliga från hela programmet. Existerar inte flera temporära objekt

av samma slag på en gång i ett system, kan det vara onödigt att definiera någon temporär identitet. Eftersom den temporära identiteten bara är lokalt definierad, kan man då istället definiera systemvariabler, som motsvarar identitetens attribut. Man behöver då inga satser för överföring mellan subroutinerna. Systemvariablerna är också kortare att skriva, eftersom en identitets attribut måste skrivas med identiteten som argument. Flera temporära identiteter av samma slag kan existera i skilda endogena händelser, eftersom de bara är definierade lokalt. Förstörs en identitet innan nästa förs över till händelsen, behöver man naturligtvis ingen kö.

I SIMSCRIPT behövs inga FORMAT-satser för utskriften. Text skrivs på samma form som den matats in. Man kan få en relativt lång utskrift per rad, eftersom varje rad motsvaras av två kort, nämligen ett vänsterhandskort och ett högerhandskort. Variabler, som man villskriva ut inuti texten, markeras med asterisker. Variablernas namn stansas på efterföljande kort. Enkel radframmatning erhålls automatiskt, medan flera raders mellanrum och ny sida markeras med stansningar i högerhandskortet. Högerhandskorten ligger i en bunt efter vänsterhandskorten. Rad- och kolonnrepetition, då man vill skriva ut matriser, fås genom att lägga in repetitionssatser i utskriften. Siffror måste stansas i särskilda positioner för att ange storleken på det fält, som skall skrivas ut. För att markera var repetitionen tar slut måste man lägga in kort, som delar utskriften i sektioner. Detta gör att det går åt många kort och att man lätt gör fel. Varje vänsterhandskort måste motsvaras av rätt högerhandskort d.v.s. ligger ett kort fel förskjuts hela utskriften. I utskrifter, som saknar fält och där man använder enkel radframmatning blir alla högerhandskort lika. SIMSCRIPT:s reportgenerator blandar då på ett enkelt och överskådligt sätt text med utskriften av variabler.

I SIMSCRIPT finns två sätt att bilda slumptal på. Flytande rektangelfördelade slumptal mellan 0 och 1 erhålls av variabeln RANDM. Man kan även få heltal slumpmässigt med hjälp av RANDI(I,J), som ger rektangelfördelade slumptal från I till J. Detta gör att man slipper multiplicera och avrunda RANDM. Slumptal, som är heltal, behövs ofta vid simulering t.ex. då man vill ge en godtycklig identitet en ny egenskap. RANDM

används mest då man vill öka tidvariabeln, TIME, som är ett flytande tal, med ett godtyckligt antal tidsenheter.

SIMSCRIPT har ett antal felutskrifter av vanlig typ. Dessa syns för den ovane en aning kryptiska, men förklaringar finns i SIMSCRIPT:s manual. Ett simuleringsprogram är svårt att teoretiskt följa i detalj, eftersom subroutinerna sker slumpmässigt. I SIMSCRIPT kan man lägga in en yttre händelse, som spårar värdena hos modellens operationer i ett visst tidsintervall och sedan skriver ut dem vid programmets slut. Vilka variabler, som skall spåras bestäms genom att en summa av koder bildas. Koderna motsvarar var för sig en typ av operation i programmet. Summan läses in till den yttre händelsen. Alla de operationer, som ingår i summan spåras. Man kan naturligtvis inte spåra stora delar av ett program, eftersom det är alltför tidskrävande, men man har en möjlighet att följa några temporära identiteters väg genom modellen.

SIMSCRIPT-programmen tar förhållandevis lång tid och är dyra att exekvera. Storleken på CPU-och totaltid visas i appendix 1 och 2, där tidsutskrifterna för tillämpningsexemplen tagits med. Det enkla tillämpningsexemplet är där ett bandjobb och är därför extra dyrt att exekvera. Det större tillämpningsexemplet ligger på en fil, som är det billigare sättet att köra program på. Se styrkort till SIMSCRIPT-program i appendix 3. Styrkorten är ganska många till antalet och måste placeras i en exakt ordning i programmet. De flesta styrkorten och varje yttre händelse, inre händelse och subroutine ger upphov till en blank sida i programmet. Detta gör att SIMSCRIPT:s programutskrift innehåller ganska mycket överflödigt papper.

De exempel på simulerade system, som vanligast förekommer i SIMSCRIPT-litteraturen är av samma typ, som det enkla tillämpningsexemplet. Dessa exempel illustrerar det utmärkande för ett SIMSCRIPT-program nämligen de temporära identiteterna, köerna samt de slumpmässigt återkommande och av vissa beslutsregler beroende händelserna. Programmen är enkla att bygga och få indata fordras till modellerna. Trots detta kan man erhålla en rad intressanta upplysningar av systemet. Parametrarna i systemen varieras och man väljer sedan de värden, som bäst uppfyller de ställda kraven. SIMSCRIPT ger på så sätt en möjlighet att enkelt konstruera billiga och lätt ändrade modeller av stora, svårmanövrerade och kanske blott fiktiva system.

Referenser

1 Gordon, Geoffry - System Simulation, Prentice-Hall INC. Englewood Cliffs, New Jersey.

2 Markowitz, Henry M., Bernard Hausner and Herbert W. Karr, SIMSCRIPT - A Simulation Programming Language, Englewood Cliffs N.J. Prentice Hall, Inc. 1963.

3 UNIVAC 1100 series SIMSCRIPT I.5 (UNIVAC:s manual).

4 Honeywell - Bull - SIMSCRIPT SYSTEM REMOTE PROCESSING DEMONSTRATION GUIDE.

Appendix 1: Enkelt tillämpningsexempel

Appendix 2: Större tillämpningsexempel

Appendix 3: Styrkort till SIMSCRIPT-program

Appendix 1: Enkelt tillämpningsexempel

Definitioner

IPART	temporär identitet
IQUAL	attribut till IPART, som anger kvalitén
ARRVT	attribut till IPART, som innehåller ankomsttiden
SQUE	attribut till IPART, som används vid uppbyggnad av kön
ARRV	"event notice", som beskriver ett föremåls ankomst
TEST	"event notice", som beskriver hur ett föremål testas

Systemvariabler

NGOOD	räknar antal bra föremål
NBAD	räknar antal dåliga föremål
NTOT	räknar hur många föremål som anlänt
FQUE	tillhör setet
LQUE	tillhör setet
NWAIT	räknar antal föremål, som väntar
NOWT	räknar antal föremål, som slipper vänta
FREE	en variabel, som sätts då arbetaren väntar
ATWT	räknar medelvärdet av väntetiden
MARRV	anger intervallet mellan föremålens ankomst
LINE	summerar antal bitar i kön
ALINE	räknar ut medelvärdet av LINE

Flödesdiagram

EXOG. EVENT START

```
CREATE ARRV
CAUSE ARRV AT TIME
     RETURN
```

EXOG. EVENT STOP

```
ALINE = FLOAT(LINE)/FLOAT(NTOT)
ATWT = ATWT/FLOAT(NBAD + NGOOD)
CALL FINAL
   STOP
   RETURN
```

ENDOG. EVENT ARRV

```
  DESTROY ARRV
  CREATE IPART
IQUAL(IPART) = RAND I (1,10)
NTOT = NTOT + 1
ARRVT(IPART) = TIME
```

FREE ≥ 0

Ja

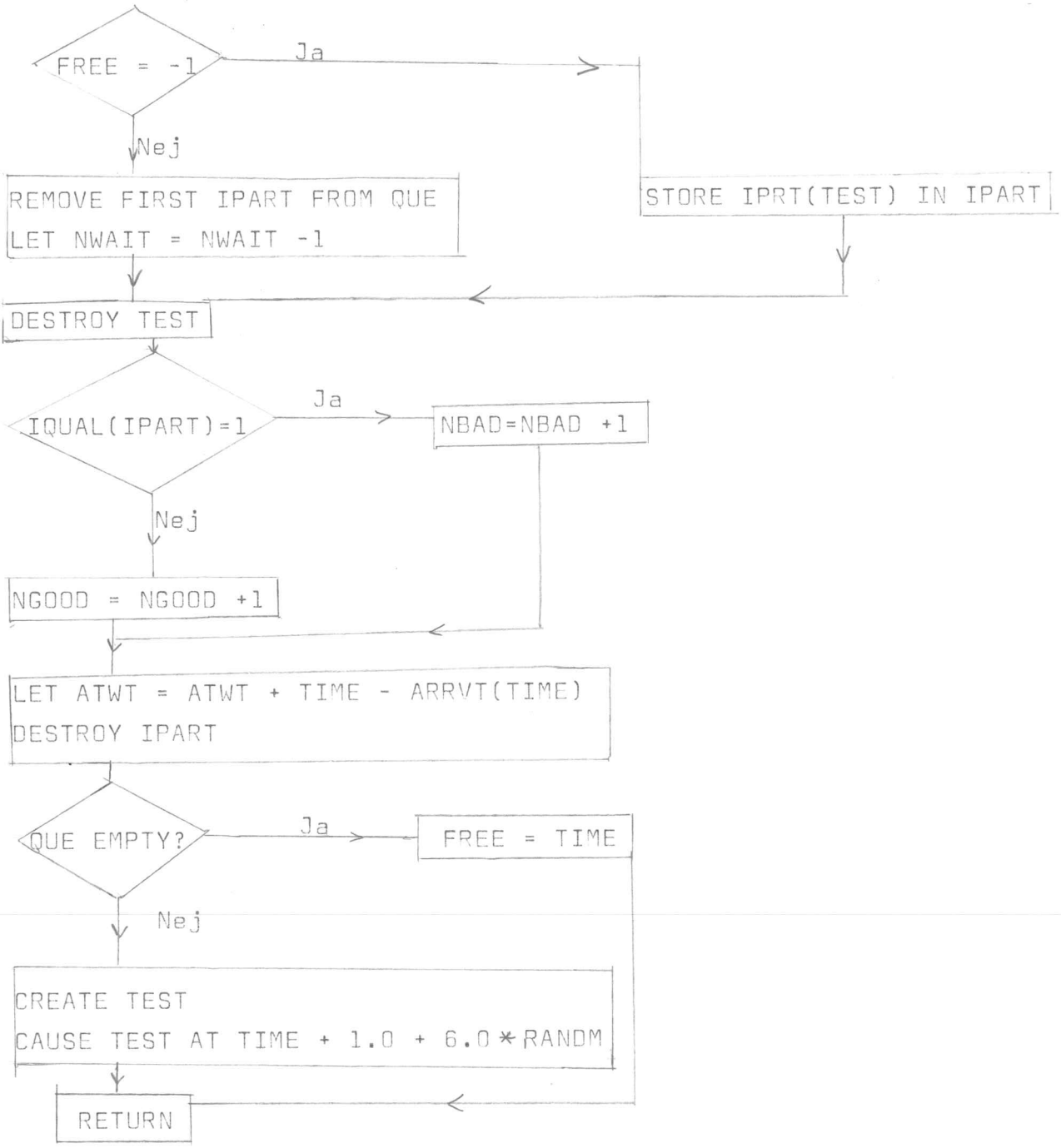
```
TWAIT = TWAIT + TIME - FREE
NOWT = NOWT + 1
FREE = - 1.0
CREATE TEST
STORE IPART IN IPRT (TEST)
CAUSE TEST AT TIME+1.0+6.0*RANDM
```

Nej

```
NWAIT = NWAIT + 1
LINE = LINE + NWAIT
FILE IPART IN QUE
```

```
CREATE ARRV
CAUSE ARRV AT TIME + MARRV
     RETURN
```

ENDOG. EVENT TEST



REPORT FINAL

Ger utskrift

Programutskrift från det enkla tillämpningsexemplet.

```
+T IPART2          T IQUAL 11/2 I   1NGOOD      I
+                  T ARRVT 12/2 F   2NBAD       I
+                  T SQUE  2    I   3NTOT       I
                                    4FQUE       I
+N ARRV 2                           5LQUE       I
+N TEST 4          N IPRT  3    I   6NWAIT      I
+                                   7NOWT       I
+                                   8FREE       F
+                                   9ATWT       F
+                                   10MARRV     F
+                                   11TWAIT     F
+                                   12LINE      I
+                                   13ALINE     F
+                                                    QUE 0 *
```

```
 EVENTS
2 EXOGENOUS
      START(1)
      STOP(2)
2 ENDOGENOUS
      ARRV
      TEST
END
```

```
EXOGENOUS EVENT STOP
LET ALINE = FLOAT(LINE)/FLOAT(NTOT)
LET ATWT = ATWT/FLOAT(NBAD + NGOOD)
CALL FINAL
STOP
RETURN
END
```

```
EXOG EVENT START
 CREATE ARRV
CAUSE ARRV AT TIME
RETURN
END
```

```
      ENDOGENOUS EVENT ARRV
      DESTROY ARRV
      CREATE IPART
      LET IQUAL(IPART) = RANDI(1,10)
      LET NTOT = NTOT + 1
      LET ARRVT(IPART) = TIME
      IF FREE GE 0,GO TO 2
      LET NWAIT = NWAIT + 1
      LET LINE = LINE + NWAIT
      FILE IPART IN QUE
      GO TO 3
2     LET TWAIT = TWAIT + TIME - FREE
      LET NOWT = NOWT + 1
      LET FREE = -1.0
      CREATE TEST
      STORE IPART IN IPRT(TEST)
      CAUSE TEST AT TIME + 1.0 + 6.0*RANDM
3     CREATE ARRV
      CAUSE ARRV AT TIME + MARRV
      RETURN
      END
```

```
      ENDOGENOUS EVENT TEST
      IF FREE EQ - 1,GO TO 4
      REMOVE FIRST IPART FROM QUE
      LET NWAIT =NWAIT - 1
      GO TO 5
4     LET FREE = -2.0
      STORE IPRT(TEST) IN IPART
5     DESTROY TEST
      IF IQUAL(IPART) EQ 1,GO TO 2
      LET NGOOD=NGOOD + 1
      GO TO 3
2     LET NBAD = NBAD + 1
3     LET ATWT = ATWT + TIME - ARRVT(IPART)
      DESTROY IPART
      IF QUE IS EMPTY,GO TO 1
      CREATE TEST
      CAUSE TEST AT TIME + 1.0 + 6.0*RANDM
      GO TO 6
1     LET FREE = TIME
6     RETURN
      END
```

```
L.                                                                  R.
      SIMSCRIPT SIMULATION OF MACHINE SYSTEM                        R. 3
L.X
      TIME BETWEEN ARRIVALS                  **.*                   R.
L.X                                                                 R.
                                             MARRV                  R.
L. X                                                                R.
      NO. OF PARTS                           ****                   R.
L.X                                                                 R.
                                             NTOT                   R.
L. X                                                                R.
      NO. OF GOOD                             ***                   R.
L.X                                                                 R.
                                             NGOOD                  R.
L. X                                                                R.
      NO. OF BAD                              ***                   R.
L.X                                                                 R.
                                             NBAD                   R.
L. X                                                                R.
      NO. OF PARTS WAITING                   ****                   R.
L.X                                                                 R.
                                             NWAIT                  R.
L. X                                                                R.
```

```
      AVERAGE TIME SPENT IN QUE
L.X                                                                  R.
      PER EXAMENED PART                 ***.*
L.X                                                                  R.
                                       ATWT
L. X                                                                 R.
      TIME WORKER IS WAITING            *****.*
L.X                                                                  R.
                                        TWAIT
L. X                                                                 R.
      AVERAGE NO. OF PARTS IN QUE       ***.*
L.X                                                                  R.
                                        ALINE
L. X                                                                 R.
      NO. OF PARTS WITH NO TIME WAITING **
L.X                                                                  R.
                                          NOWT
L. X                                                                 R.
      CLOCK TIME                        ****.*
L.X                                                                  R.
                                        TIME
L. X                                                                 R.
      END                                                 END
L.                                                                   R.
```

SIMSCRIPT SIMULATION OF MACHINE SYSTEM

TIME BETWEEN ARRIVALS	4.6
NO. OF PARTS	105
NO. OF GOOD	91
NO. OF BAD	13
NO. OF PARTS WAITING	0
AVERAGE TIME SPENT IN QUE PER EXAMENED PART	5.6
TIME WORKER IS WAITING	56.0
AVERAGE NO. OF PARTS IN QUE	1.1
NO. OF PARTS WITH NO TIME WAITING	41
CLOCK TIME	480.

RUNID SIM ACCT 208461 PROJ SIM PRIO M&L

LOAD 2573 2/0 2573 -1 SIM

START 10:38:30 SEP 28,1973 FIN 10:41:18 SEP 28,1973 QUEUED 00:04

IMAGES READ 956 PAGES 21 IMAGES PUNCHED 0

I/O BY GROUP:	DRUM	DISC	TAPE
MILLISECONDS:	12140	3058	4561

DELKOSTN.: 5.84 CPU; 3.60 KÄRNMODUL; 0.00 REALTID
5.00 BANDMONT+PLOTT+FÄRGB+REMSOR; 0.00 UPPKOPPL; 0.00 TEMP FIL
0.00 KORT UT; 7.79 KORT IN; 4.75 SIDOR

PRIS KR 26.97 TIME: TOTAL:00:01:10.746 CC/ER:00:00:25.901 CPU:00:00:08.891 I/O:00:00:19.760 MEM:00:00:16.193 WAIT:00:00:00.120

Appendix 2: Större tillämpningsexempel

Definitioner

TPER är en "event notice" och anger att det finns en inre händelse med detta namn.

De övriga definitionerna är systemvariabler d v s räknare och fält, som sätts initialt eller får värden av programmet.

ITNT anger antal tentor

KBTG anger antal betyg

NRUN anger hur många elever som ingår i kursen

SVKT anger summan av vikterna för tentorna

LPLUS räknar antal lyckade plussningar

NBTOT räknar hur många elever som fått ett visst betyg i ett visst ämne

KPVLV räknar kumulativt antal godkända elever i procent, i ett visst ämne

SBPER anger antal kolonner i fältet NBTOT m.fl.

BTGL anger hur betygen är fördelade för de medelbra eleverna

BTGL 1 anger hur betygen är fördelade för de dåliga eleverna

BTGL 3 anger hur betygen är fördelade för de bra eleverna

PBTOT anger NBTOT i procent av samtliga elever

ATNT räknar ut ett medelvärde av hur många gånger en elev tenterat under de två första åren (i procent)

MTNT räknar samma som NTNT, fast i procent

IGODK räknar antal elever med inga godkända tentor

FIGDK räknar IGODK i procent

KLAR håller reda på elevens betyg

NPLUS räknar plussare

PRT anger hur stor del ett visst antal tentor utgör av samtliga (i procent)

NTNT räknar hur många elever, som klarat ett visst antal tentor

KPER anger antal tentamensperioder

SKVL anger antal kolonner i fältet NPKV (nedan)

SEKT anger sektion

KVOT anger storleken på förskjutningen mellan BTGL1, BTGL3 och BTGL

NAMN anger tentans namn

VIKT anger ämnets storlek, 1 = litet, 2 = stort

NVTNT räknar hur många elever, som klarat ett visst antal tentor, då tentorna viktats

VPRT anger hur stor del ett visst antal tentor utgör av samtliga efter viktning

KVAL anger elevens kvalité 1 = dålig, 2 = medel, 3 = bra

NPKV räknar hur många elever med en viss kvalité, som blivit godkända i ett visst ämne

PELV räknar om NTNT i procent

PVELV räknar om NVTNT i procent

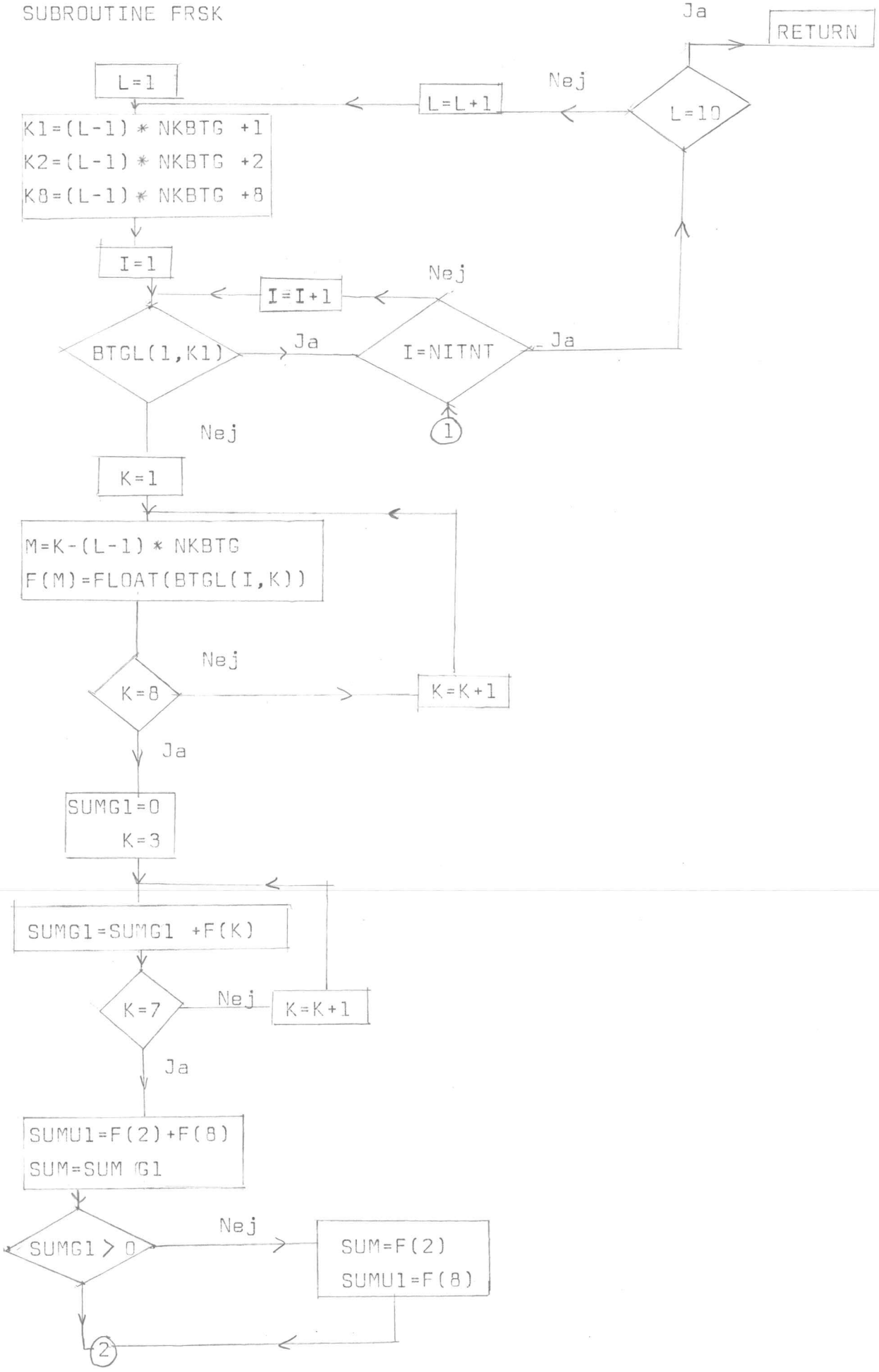
KV räknar antal elever av varje kvalité

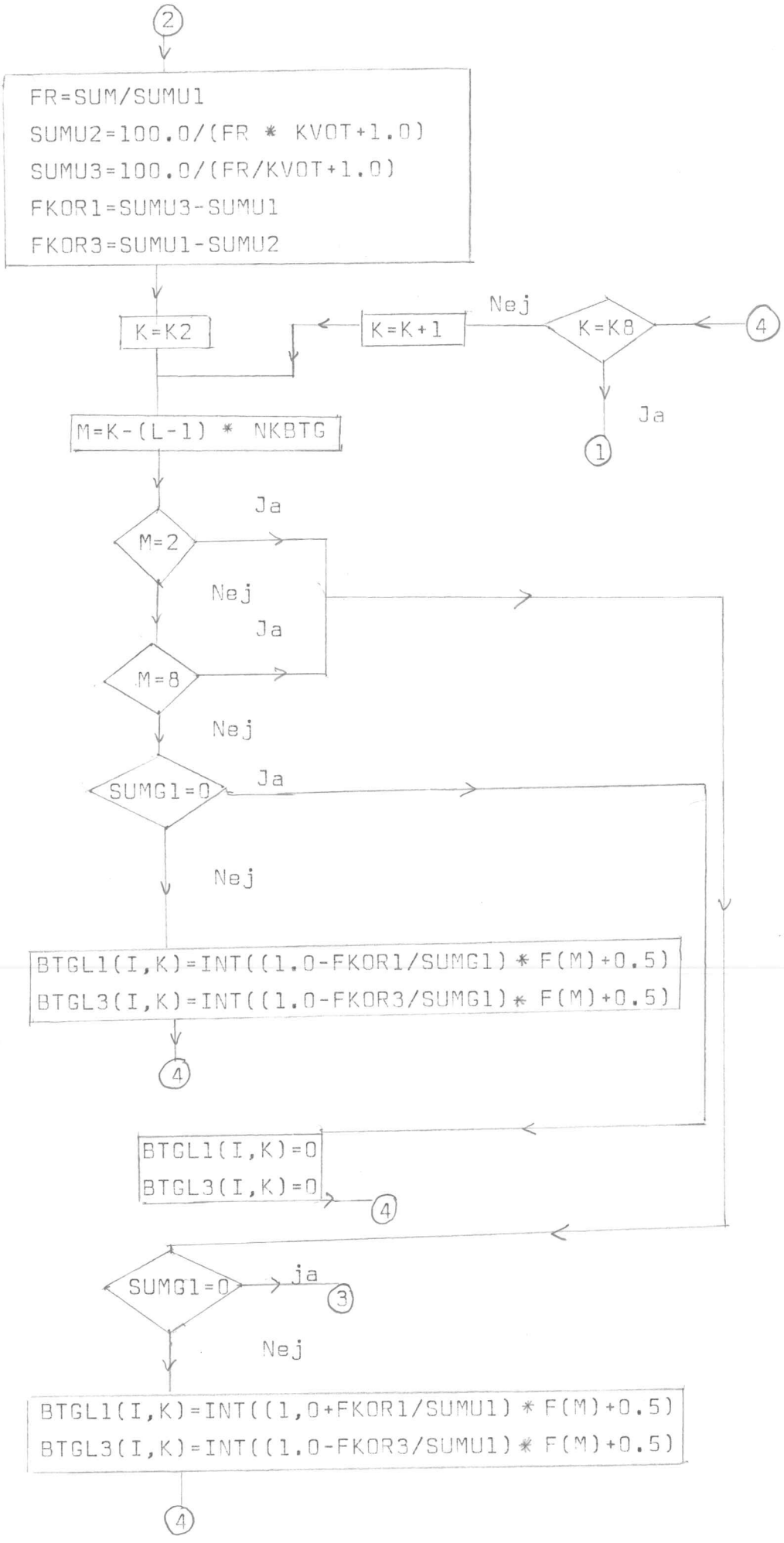
FNPKV räknar ut NPKV i procent av KV för varje kvalité och ämne

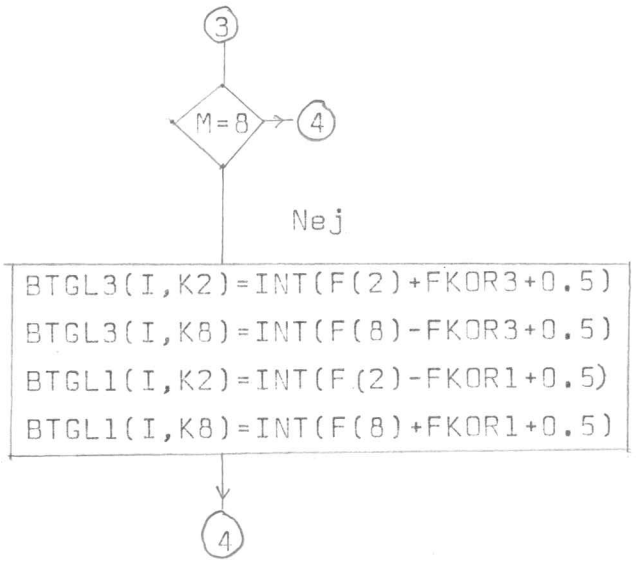
FRD bestämmer hur stor procent av mängden som är bra och dåliga

Nedan följer flödesdiagram över det större tillämpningsexemplet.

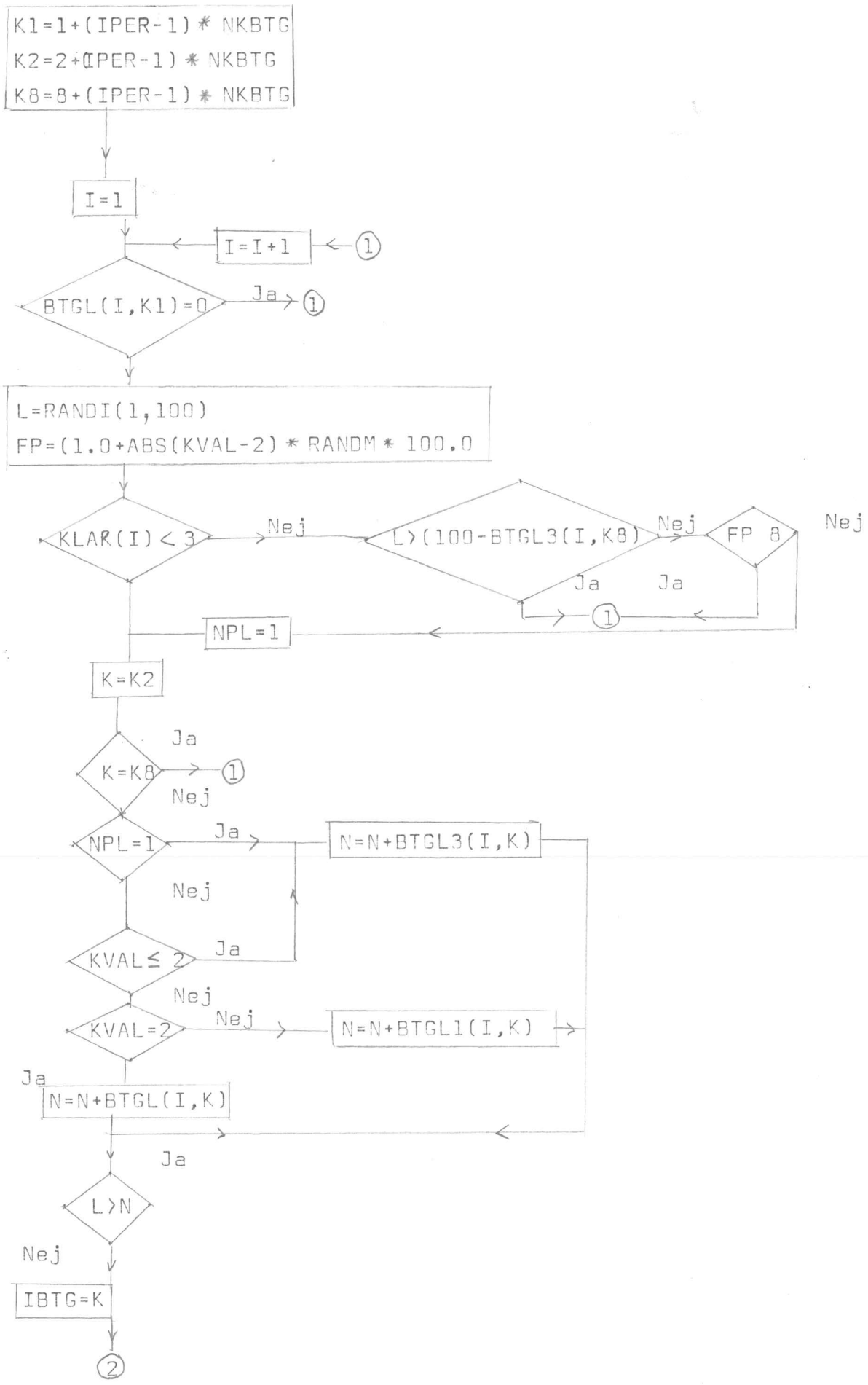
SUBROUTINE FRSK

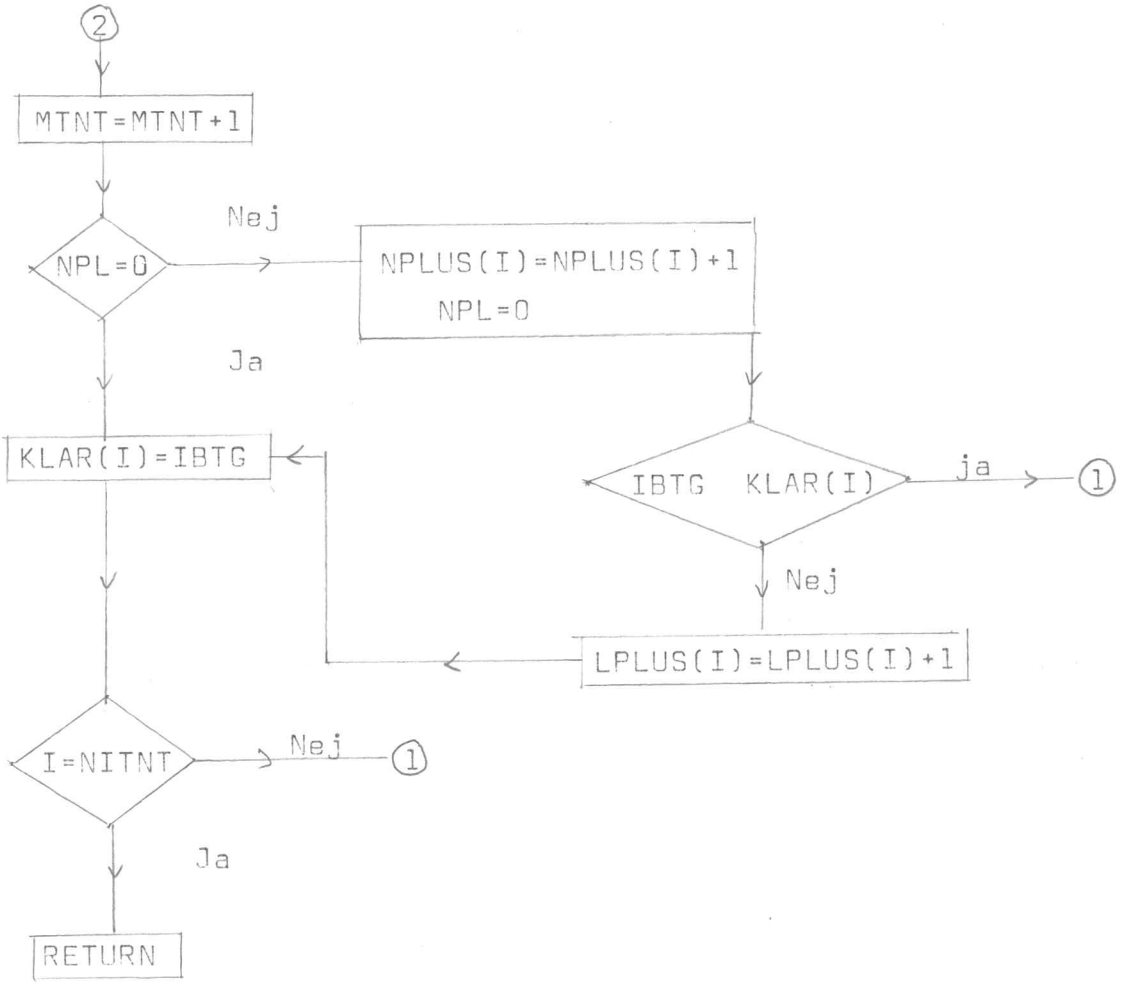



2
FR=SUM/SUMU1
SUMU2=100.0/(FR * KVOT+1.0)
SUMU3=100.0/(FR/KVOT+1.0)
FKOR1=SUMU3-SUMU1
FKOR3=SUMU1-SUMU2
K=K2
K=K+1
Nej
K=K8
4
Ja
1
M=K-(L-1) * NKBTG
Ja
M=2
Nej
Ja
M=8
Nej
SUMG1=0
Ja
Nej
BTGL1(I,K)=INT((1.0-FKOR1/SUMG1) * F(M)+0.5)
BTGL3(I,K)=INT((1.0-FKOR3/SUMG1) * F(M)+0.5)
4
BTGL1(I,K)=0
BTGL3(I,K)=0
4
SUMG1=0
ja
3
Nej
BTGL1(I,K)=INT((1,0+FKOR1/SUMU1) * F(M)+0.5)
BTGL3(I,K)=INT((1.0-FKOR3/SUMU1) * F(M)+0.5)
4

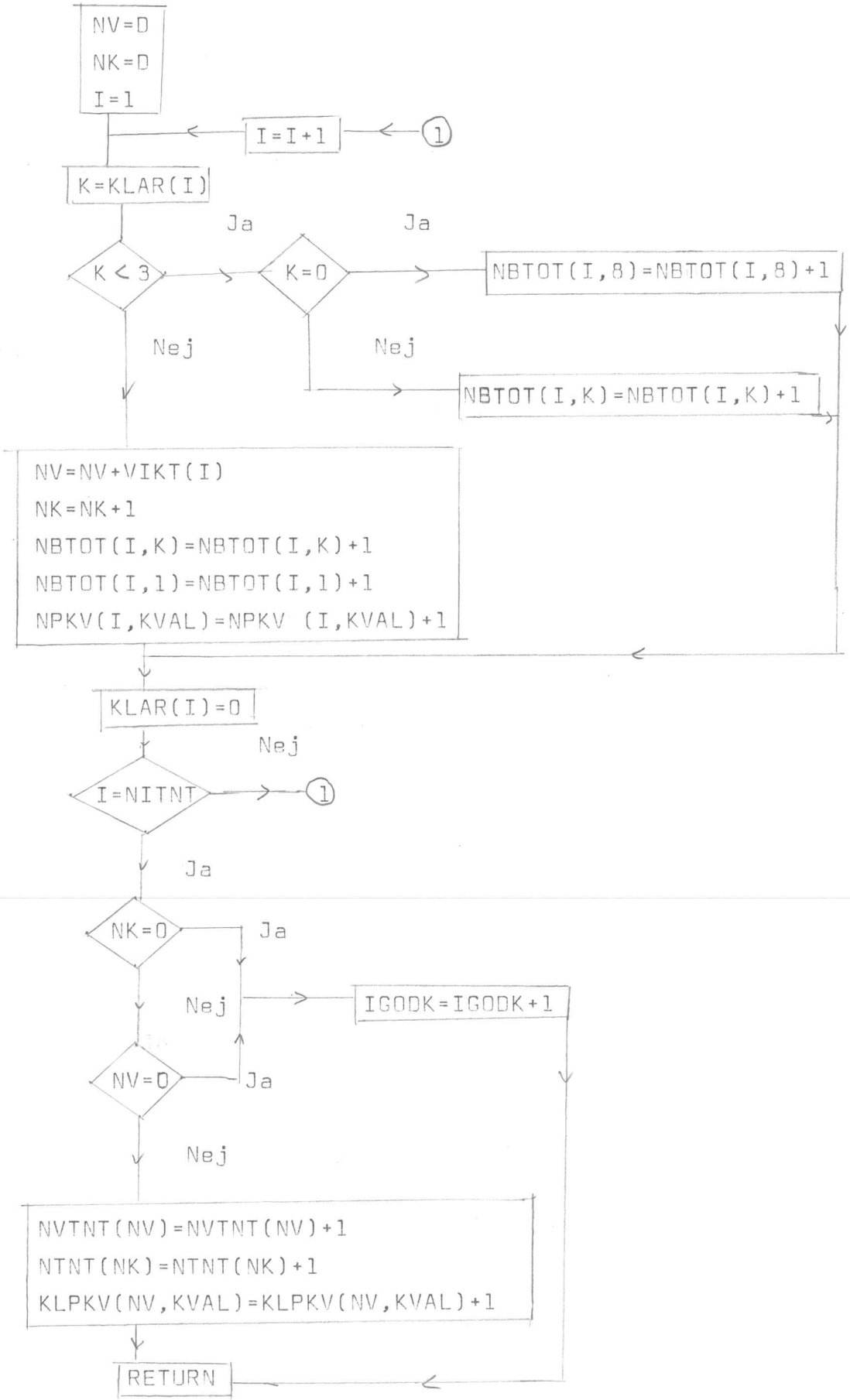

3
M=8
4
Nej
BTGL3(I,K2)=INT(F(2)+FKOR3+0.5)
BTGL3(I,K8)=INT(F(8)-FKOR3+0.5)
BTGL1(I,K2)=INT(F(2)-FKOR1+0.5)
BTGL1(I,K8)=INT(F(8)+FKOR1+0.5)
4

SUBROUTINE NBTG(IPER)

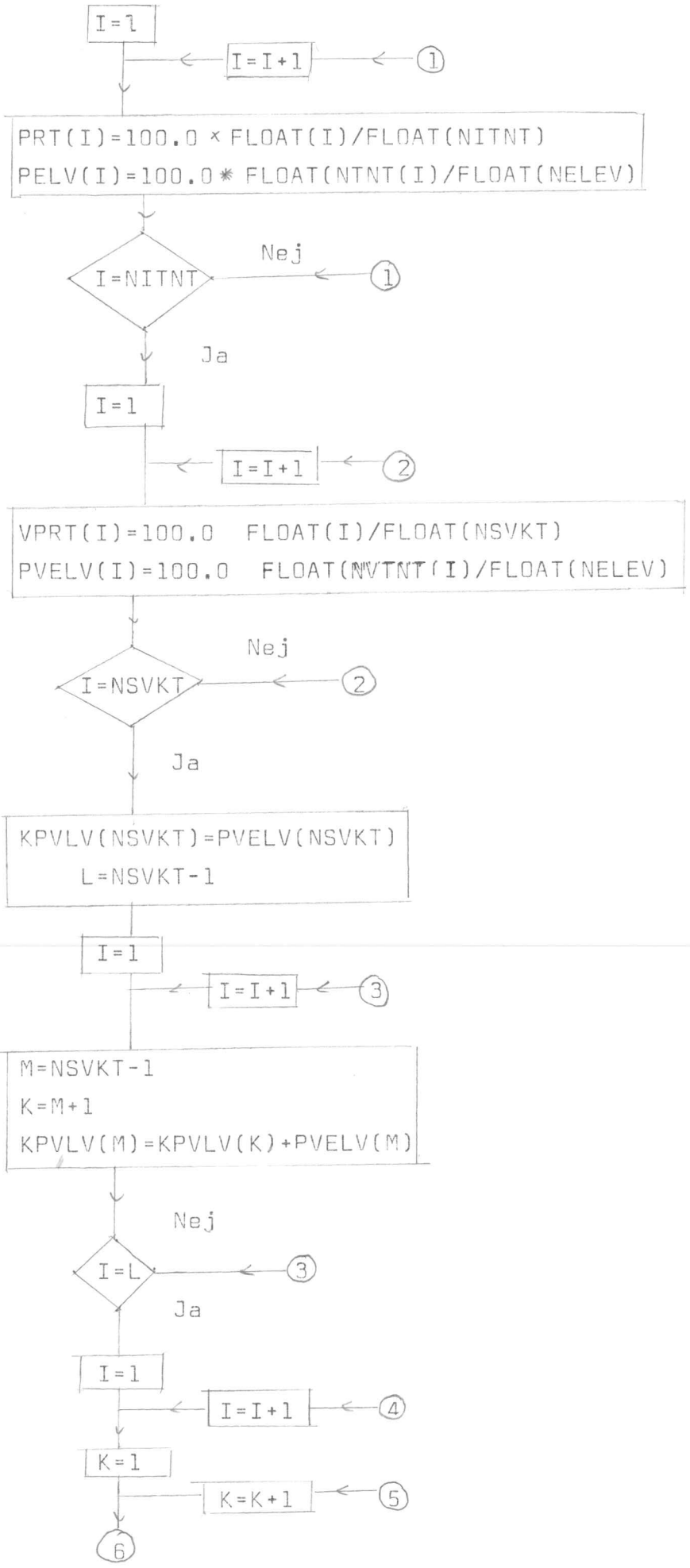


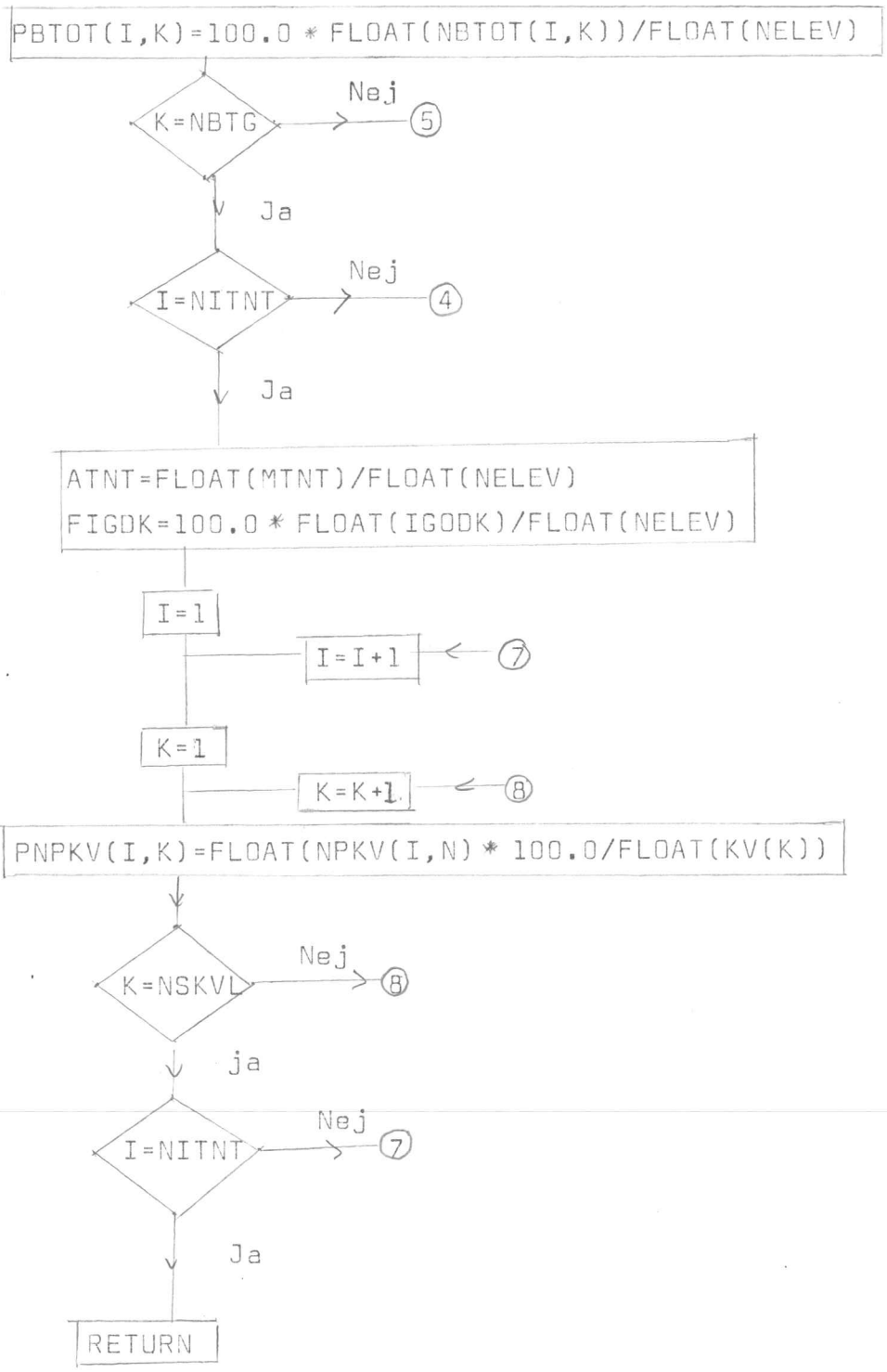

2
MTNT=MTNT+1
NPL=0
Nej
NPLUS(I)=NPLUS(I)+1
NPL=0
Ja
KLAR(I)=IBTG
IBTG KLAR(I)
ja
1
Nej
LPLUS(I)=LPLUS(I)+1
I=NITNT
Nej
1
Ja
RETURN

SUBROUTINE KRES

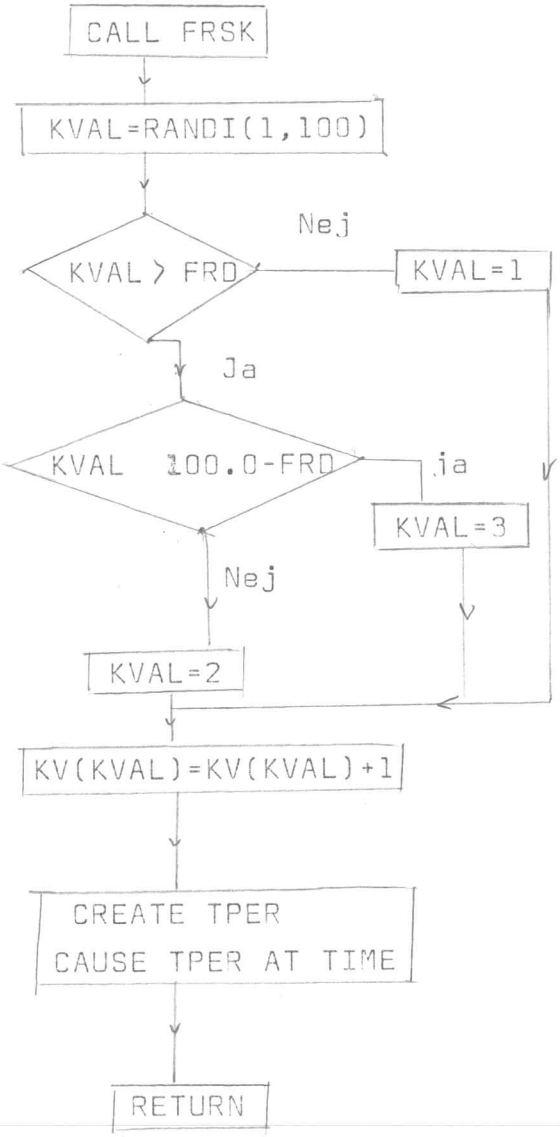


SUBROUTINE PROC

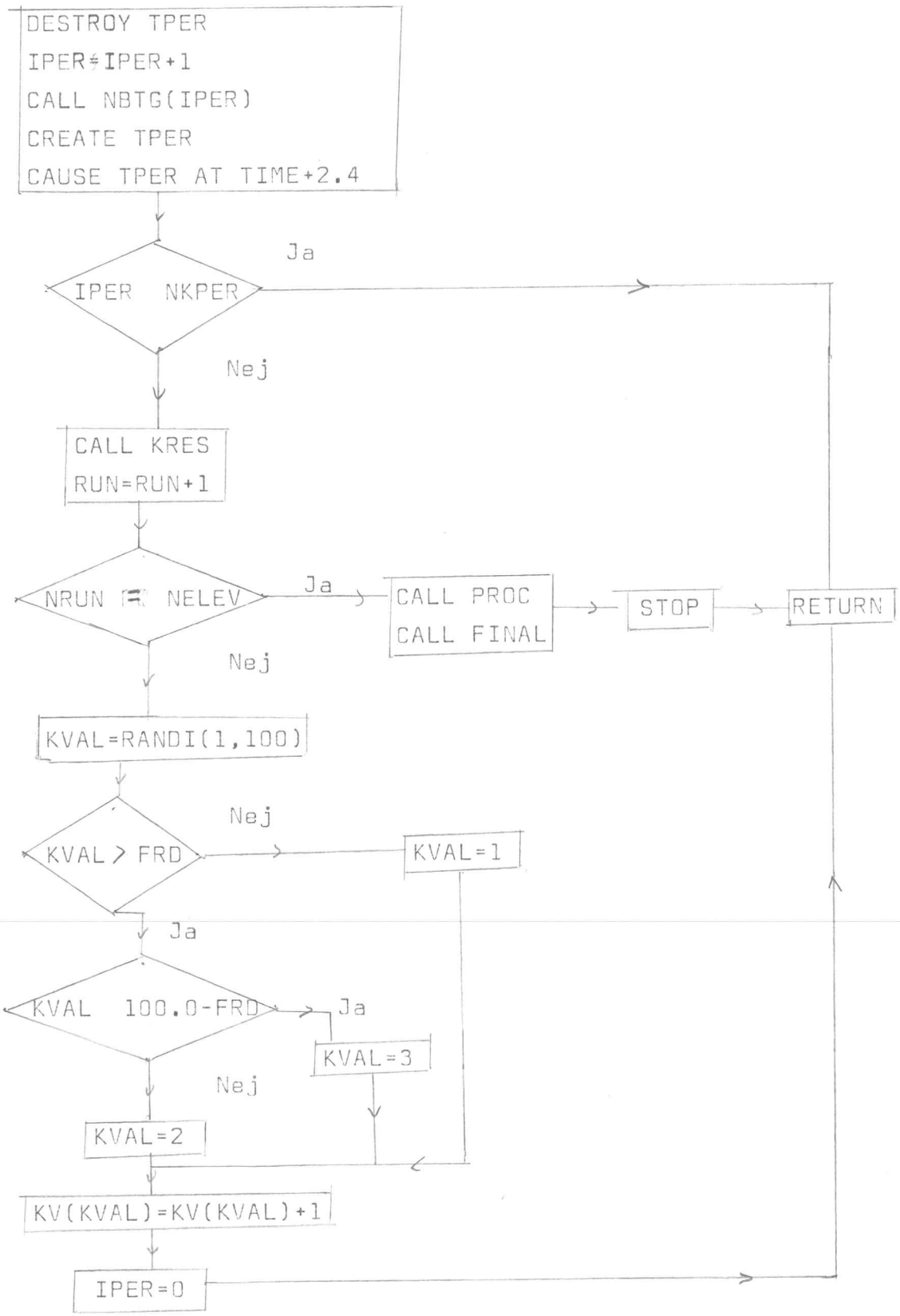



PBTOT(I,K)=100.0 * FLOAT(NBTOT(I,K))/FLOAT(NELEV)
K=NBTG
Nej
5
Ja
I=NITNT
Nej
4
Ja
ATNT=FLOAT(MTNT)/FLOAT(NELEV)
FIGDK=100.0 * FLOAT(IGODK)/FLOAT(NELEV)
I=1
I=I+1
7
K=1
K=K+1
8
PNPKV(I,K)=FLOAT(NPKV(I,N) * 100.0/FLOAT(KV(K))
K=NSKVL
Nej
8
ja
I=NITNT
Nej
7
Ja
RETURN

EXOG. EVENT TPER



ENDOG. EVENT TPER



Programutskrift från det större tillämpningsexemplet

```
+N TPER 2                          1ITNT   E
+                                  2KBTG   E
+                                  3NRUN       I
+                                  4NELEV      I
+                                  5SVKT   E
+                                  6LPLUS  1   I
+                                  7NBTOT  2   I
+                                  8KPVLV  1   F
+                                  9SBPER  E
+                                 10BTGL   2   I
+                                 11BTGL1  2   I
+                                 12BTGL3  2   I
+                                 13PBTOT  2   F
+                                 14ATNT       F
+                                 15MTNT       I
+                                 16IGODK      I
+                                 17FIGDK      F
+                                 18KLAR   1   I
+                                 19NPLUS  1   I
+                                 20PRT    1   F
+                                 21NTNT   1   I
+                                 22KPER   E
+                                 23SKVL   E
+                                 24SEKT       I
+                                 25KVOT       F
+                                 26NAMN   1   I
+                                 27VIKT   1   I
+                                 28NVTNT  1   I
+                                 29VPRT   1   F
+                                 30KVAL       I
+                                 31NPKV   2   I
+                                 32PELV   1   F
+                                 33PVELV  1   F
+                                 34KV     1   I
+                                 35PNPKV  2   F
+                                 36KLPKV  2   I
+                                 37FRD        F
```

```
EVENTS
1 EXOG
START(1)
1 ENDOG
TPER
END
```

```
      SUBROUTINE FRSK
C     FORSKJUTER DEN INMATADE BETYGSFORDELNINGEN(BTGL)
C     I EN BRA FORDELNING(BTGL3) OCH EN DALIG(BTGL1)
      DIMENSION F(8)
      DO,FOR L=(1)(10)
      LET K1 = (L-1)*NKBTG + 1
      LET K2 = (L-1)*NKBTG + 2
      LET K8 = (L-1)*NKBTG + 8
      DO,FOR I = (1)(NITNT)
      IF BTGL(I,K1) EQ 0,GO TO 1
      DO ,FOR K = (K1)(K8)
      LET M = K - (L -1)*NKBTG
      LET F(M) = FLOAT(BTGL(I,K))
      REPEAT
      LET SUMG1 = 0
      DO ,FOR K = (3)(7)
      LET SUMG1 = SUMG1 +  F(K)
      REPEAT
      LET SUM = SUMG1
      LET SUMU1 = F(2) + F(8)
      IF SUMG1 GR 0,GO TO 2
      LET SUM = F(2)
      LET SUMU1 =F(8)
2     LET FR = SUM/SUMU1
      LET SUMU2 = 100.0/(FR*KVOT + 1.0)
      LET SUMU3 = 100.0/(FR/KVOT + 1.0)
      LET FKOR1 = SUMU3 - SUMU1
      LET FKOR3 = SUMU1 - SUMU2
      DO,FOR K =(K2)(K8)
      LET M = K - (L - 1)*NKBTG
      IF M EQ 2,GO TO 3
      IF M EQ 8,GO TO 3
      IF SUMG1 EQ 0,GO TO 4
      LET BTGL1(I,K) = INT((1.0 - FKOR1/SUMG1)*F(M) + 0.5)
      LET BTGL3(I,K) = INT((1.0 + FKOR3/SUMG1)*F(M) + 0.5)
      GO TO 6
4     LET BTGL1(I,K) = 0
      LET BTGL3(I,K) = 0
      GO TO 6
3     IF SUMG1 EQ 0,GO TO 5
      LET BTGL1(I,K) = INT((1.0 + FKOR1/SUMU1)*F(M) + 0.5)
      LET BTGL3(I,K) = INT((1.0 - FKOR3/SUMU1)*F(M) + 0.5)
      GO TO 6
5     IF M EQ 8,GO TO 6
      LET BTGL3(I,K2) = INT(F(2)  + FKOR3 + 0.5)
      LET BTGL3(I,K8) = INT(F(8)  - FKOR3 + 0.5)
      LET BTGL1(I,K2) = INT(F(2)  - FKOR1 + 0.5)
      LET BTGL1(I,K8) = INT(F(8)  + FKOR1 + 0.5)
6     REPEAT
1     REPEAT
      REPEAT
      RETURN
      END
```

```
      SUBROUTINE NBTG(IPER)
C     RAKNAR UT ELEVENS BETYG
C     ELEVENS KVALITE = 1 MEDFOR BETYG ENL. BGTL1, KVALITE = 2 ENL. BTGL
C      OCH KVALITE = 3 ENL. BGTL3
C     ANTAL PLUSSNINGAR BESTAMS SOM ETT ANTAL PROCENT AV ETT SLUMPTAL ,
C     GENERERAT SOM FUNKTION AV ELEVENS KVALITE
      LET K1 = 1 + (IPER-1)*NKBTG
      LET K2 = 2 + (IPER - 1)*NKBTG
      LET K8 = 8 + (IPER-1)*NKBTG
      DO,FOR I = (1)(NITNT)
      IF BTGL(I,K1) EQ 0 ,GO TO 2
      LET L = RANDI(1,100)
      LET FP=(1.0 + ABS(KVAL - 2))*RANDM*100.0
      LET N = 0
      IF KLAR(I) LS 3,GO TO 4
      IF L GR (100 - BTGL3(I,K8)),GO TO 2
      IF FP GR 8,GO TO 2
      LET NPL = 1
4     DO,FOR K = (K2)(K8)
      IF K EQ K8,GO TO 2
      IF NPL EQ 1 ,GO TO 5
      IF KVAL LE 2,GO TO 11
5     LET N = N + BTGL3(I,K)
      GO TO 12
11    IF KVAL EQ 2,GO TO 13
      LET N = N + BTGL1(I,K)
      GO TO 12
13    LET N = N + BTGL(I,K)
12    IF L GR N,GO TO 1
      LET IBTG = K - (IPER - 1)*NKBTG
      GO TO 3
1     REPEAT
3     LET MTNT = MTNT + 1
      IF NPL EQ 0,GO TO 10
      LET NPL = 0
      LET NPLUS(I) = NPLUS(I) + 1
      IF IBTG LE KLAR(I),GO TO 2
      LET LPLUS(I) = LPLUS(I) + 1
10    LET KLAR(I) = IBTG
2     REPEAT
      RETURN
      END
```

```
      SUBROUTINE KRES
C     RAKNAR UT HUR MÅNGA ELEVER,SOM FÅTT ETT VISST RESULTAT
      LET NK = 0
      LET NV = 0
      DO,FOR I=(1)(NITNT)
      LET K= KLAR(I)
      IF K LS 3,GO TO 1
      LET NV = NV + VIKT(I)
      LET NK = NK + 1
      LET NBTOT(I,K) =NBTOT(I,K) + 1
      LET NBTOT(I,1) = NBTOT(I,1) + 1
      LET NPKV(I,KVAL) = NPKV(I,KVAL) + 1
      GO TO 3
1     IF K EQ 0,GO TO 2
      LET NBTOT(I,K) =NBTOT(I,K) + 1
      GO TO 3
2     LET NBTOT(I,8) = NBTOT(I,8) + 1
3     LET KLAR(I) = 0
      REPEAT
      IF NK EQ 0,GO TO 4
      IF NV EQ 0,GO TO 4
      LET NVTNT(NV) = NVTNT(NV) + 1
      LET KLPKV(NV,KVAL) =KLPKV(NV,KVAL) + 1
      LET NTNT(NK) = NTNT(NK) + 1
      GO TO 5
4     LET IGODK = IGODK + 1
5     RETURN
      END


      SUBROUTINE PROC
C      RAKNAR ANTAL ELEVER MED ETT VISST RESULTAT I PROCENT AV SAMTLIGA
C      RAKNAR UT HUR STOR PROCENT ETT VISST ANTAL TENTOR UTGÖR
C     AV SAMTLIGA( MED OCH UTAN VIKTNING)
      DO,FOR I=(1)(NITNT)
      LET PRT(I) = 100.0*FLOAT(I)/FLOAT(NITNT)
      LET PELV(I) = 100.0*FLOAT(NTNT(I))/FLOAT(NELEV)
      REPEAT
      DO,FOR I=(1)(NSVKT)
      LET VPRT(I) = 100.0*FLOAT(I)/FLOAT(NSVKT)
      LET PVELV(I) = 100.0*FLOAT(NVTNT(I))/FLOAT(NELEV)
      REPEAT
      LET KPVLV(NSVKT) = PVELV(NSVKT)
      LET L = NSVKT - 1
      DO,FOR I = (1)(L)
      LET M = NSVKT - I
      LET K = M + 1
      LET KPVLV(M) = KPVLV(K) + PVELV(M)
      REPEAT
      DO,FOR I = (1)(NITNT)
      DO,FOR K = (1)(NKBTG)
      LET PBTOT(I,K) = 100.0*FLOAT(NBTOT(I,K))/FLOAT(NELEV)
      REPEAT
      REPEAT
      LET ATNT = FLOAT(MTNT)/ FLOAT(NELEV)
      LET FIGDK = 100.0*FLOAT(IGODK)/FLOAT(NELEV)
      DO,FOR I =(1)(NITNT)
      DO,FOR K =(1)(NSKVL)
      LET PNPKV(I,K) = FLOAT(NPKV(I,K))*100.0/FLOAT(KV(K))
      REPEAT
      REPEAT
      RETURN
      END
```

```
      EXOG EVENT START
C     GER ELEV NR 1 EN KVALITE
C     STARTAR UPP TPER
      CALL FRSK
      LET KVAL = RANDI(1,100)
      IF KVAL GR FRD,GO TO 3
      LET KVAL =1
      GO TO 1
3     IF KVAL GR (100.0 - FRD),GO TO 4
      LET KVAL = 2
      GO TO 1
4     LET KVAL = 3
1     LET KV(KVAL) = KV(KVAL) + 1
      CREATE TPER
      CAUSE TPER AT TIME
      RETURN
      END



      ENDOG EVENT TPER
C     RAKNAR PERIODER OCH ELEVER
C     GER ELEVEN EN KVALITE
      DESTROY TPER
      LET IPER = IPER + 1
      CALL NBTG(IPER)
      CREATE TPER
      CAUSE TPER AT TIME + 2.4
      IF IPER LS NKPER,GO TO 1
      CALL KRES
      LET NRUN = NRUN + 1
      IF NRUN EQ NELEV,GO TO 5
      LET KVAL = RANDI(1,100)
      IF KVAL GR FRD,GO TO 3
      LET KVAL =1
      GO TO 2
3     IF KVAL GR (100.0 - FRD),GO TO 4
      LET KVAL = 2
      GO TO 2
4     LET KVAL = 3
2     LET KV(KVAL) = KV(KVAL) + 1
      LET IPER = 0
      GO TO 1
5     CALL PROC
      CALL FINAL
      STOP
1     RETURN
      END
```

```
REPORT FINAL
          245.    L.
                  SIMSCRIPT SIMULERING AV TENTAMENSSYSTEM FOR *A*
          246.    L.X
                                                                SEKT
          247.    L. X
                  ANTAL ELEVER                                  ****
          248.    L.X
                                                                NELEV
          249.    L. X
                  ANTAL BRA ELEVER = ANTAL DÅLIGA ELEVER = **.*%
          250.    L.X
                                                                  FRD
          251.    L. X
                  KVOT                                        ***.**
          252.    L.X
                                                                KVOT
          253.    L. X
                  ANTAL TENTAMENSTILLFALLEN I MEDEL PER ELEV **.**
          254.    L.X
                                                              ATNT
          255.    L. X
                  ANTAL ELEVER MED INGA GODKANDA TENTOR ( I % )
          256.    L.X
                                                              FIGDK
          257.    L. X

          258.    L.X
                  ANTAL KLARADE TENTOR      ( I %)     ANTAL ELEVER
          259.    L.X
                  (TENTORNA EJ VIKTADE EFTER IBORDES STORLEK)
          260.    L.X
                            **              **.**          ***
          261.    L.X
                            K               PRT(K)       NTNT(K)
          262.    L. X
FOR K=(1)(NITNT)
          263.    L.  X

          264.    L.X
FOR EACH SKVL K
          265.    L.    3
                  ANTAL KLARADE TENTOR I %       ANTAL ELEVER
          266.    L.X
                  (TENTORNA VIKTADE EFTER INBORDES STORLEK)
          267.    L.X

          268.    L.X

          269.    L. X
                            **.**                          ***
          270.    L.X
                            VPRT(L)                      NVTNT(L)
          271.    L. X
FOR L =(1)(NSVKT)
          272.    L.  X

          273.    L.X
FOR EACH KBTG J
          274.    L.     8
                       GODKD   UDRKD   BTG 3   BTG 4   BIG 5
          275.    L.X
                       (ANTAL ELEVER I PROCENT)
```

```
                                                    R.                  299.
                                                    R. 2                300.
                                                    R.                  301.
                                                    R.                  302.
                                                    R.                  303.
                                                    R.                  304.
                                                    R.                  305.
                                                    R.                  306.
                                                    R.                  307.
                                                    R.                  308.
                                                    R.                  309.
**.*
                                                    R.                  310.
                                                    R.                  311.
                                                    R. 2XX              312.
  ( I % )
                                                    R. 1                313.
                                                    R. 2                314.
  **.**
                                                    R.                  315.
 PELV(K)
                                                    R.                  316.
                                                    R.                  317.
                                                    R. 2XX              318.
                                                    R.                  319.
( I % )  (KUMULERAT ANTAL I %)  ELEVERNA FÖRDELADE PÅ KVALITE
                                                    R. 1                320.
                          KVALITE    1     2     3
                                                    R. 1                321.
                          SUMMA      **    **    **
                                                    R. 2                322.
                                                     3(KV(K))
                                                    R.                  323.
**.**          ****.*                 ***   ***   ***
                                                    R.                  324.
PVELV(L)       KPVLV(L)                    3(KLPKV(L,K))
                                                    R.                  325.
                                                    R.                  326.
                                                    R. 3XX              327.
                                                    R.                  328.
 BTG 6   BTG 7   EJTNT
                                                    R. 1                329.
```

```
            276.     L.X
              ***A**                **.*     **.*     **.*     **.*     **.*
            277.     L.X
                NAMN(I)                      8(PBTOT(I,J))
            278.     L. X
FOR I = (1)(NITNT)
            279.     L.  X

            280.     L.X
FOR EACH KBTG J
            281.     L.     8
                              GODK     UDRKD    BGT 3    BTG 4    BTG 5
            282.     L.X
              ***A**                 ***      ***      ***      ***      ***
            283.     L.X
              NAMN(I)                      8(NBTOT(I,J))
            284.     L. X
FOR I=(1)(NITNT)
            285.     L.  X

            286.     L.X
FOR EACH SKVL N
            287.     L.     3
                     GODKANDA ENLIGT   KVAL1    KVAL2    KVAL3
            288.     L.X
                     (RÄKNAT I PROCENT AV SAMTLIGA MED RESP. KVALITE)
            289.     L.X
                           ***A**          **.*     **.*      **.*
            290.     L.X
                           NAMN(L)           3(PNPKV(L,N))
            291.     L. X
FOR L =(1)(NITNT)
            292.     L.  X

            293.     L.X
                                      ANTAL PLUSSARE  LYCKADE PLUSSNINGAR
            294.     L.X
                           ***A**          ***                  ***
            295.     L.X
              NAMN(I)          NPLUS(I)                  LPLUS(I)
            296.     L. X
 FOR I =(1)(NITNT)
            297.     L.  X
END
            298.     L.
```

```
**.*      **.*   **.*

BGT 6     BTG 7  EJTNT
***       ***    ***

END
```

```
R. 2      330.
R. 1      331.
R.        332.
R.        333.
R. 3XX    334.
R.        335.
R. 3      336.
R. 1      337.
R.        338.
R.        339.
R. 3XX    340.
R.        341.
R. 1      342.
R. 3      343.
R.        344.
R.        345.
R.        346.
R. 3XX    347.
R. 2      348.
R.        349.
R.        350.
R.        351.
R.        352.
```

SIMSCRIPT SIMULERING AV TENTAMENSSYSTEM FOR E

ANTAL ELEVER 114
ANTAL BRA ELEVER = ANTAL DRLIGA ELEVER = 55.3%
KVOT 4.50
ANTAL TENTAMENSTILLFALLEN I MEDEL PER ELEV 16.04
ANTAL ELEVER MED INGA GODKANDA TENTOR (I %) 0.

ANTAL KLARADE TENTOR	(I %)	ANTAL ELEVER	(I %)

(TENTORNA EJ VIKTADE EFTER IBORDES STORLEK)

1	5.88	3	2.63
2	11.76	4	3.51
3	17.65	9	7.89
4	23.53	7	6.14
5	29.41	9	7.89
6	35.29	7	6.14
7	41.18	4	3.51
8	47.06	3	2.63
9	52.94	6	5.26
10	58.82	10	8.77
11	64.71	7	6.14
12	70.59	3	2.63
13	76.47	4	3.51
14	82.35	3	2.63
15	88.24	15	13.16
16	94.12	17	14.91
17	100.	3	2.63

ANTAL KLARADE TENTOR I % (TENTORNA VIKTADE EFTER INBORDES STORLEK)	ANTAL ELEVER	(I %)	(KUMULERAT ANTAL I %)	ELEVERNA FÖRDELADE PÅ KVALITÉ		
				KVALITÉ 1	2	3
				SUMMA 37	37	40
3.85	2	1.75	100.0	2	0	0
7.69	1	0.88	98.2	1	0	0
11.54	3	2.63	97.4	3	0	0
15.38	6	5.26	94.7	6	0	0
19.23	6	5.26	89.5	6	0	0
23.08	4	5.51	84.2	4	0	0
26.92	9	7.89	80.7	8	1	0
30.77	7	6.14	72.8	5	2	0
34.62	2	1.75	66.7	1	1	0
38.46	2	1.75	64.9	1	1	0
42.31	0	0.	63.2	0	0	0
46.15	4	5.51	63.2	0	4	0
50.	5	4.39	59.6	0	5	0
53.85	2	1.75	55.3	0	2	0
57.69	7	6.14	53.5	0	7	0
61.54	6	5.26	47.4	0	6	0
65.38	3	2.63	42.1	0	3	0
69.23	3	2.63	39.5	0	3	0
73.08	4	5.51	36.8	0	2	2
76.92	1	0.88	33.3	0	0	1
80.77	2	1.75	32.5	0	0	2
84.62	9	7.89	30.7	0	0	9
88.46	6	5.26	22.8	0	0	6
92.31	15	13.16	17.5	0	0	15
96.15	2	1.75	4.4	0	0	2
100.	3	2.63	2.6	0	0	3

	GODKD	UDRKD	BTG 3	BTG 4	BTG 5	BTG 6	BTG 7	EJTNT
	(ANTAL ELEVER I PROCENT)							
LIN A	71.1	26.3	23.7	31.6	12.3	3.5	0.	2.6
FSK I	73.7	16.7	40.4	24.6	8.8	0.	0.	9.6
ANL 1	76.3	17.5	46.5	18.4	10.5	0.9	0.	6.1
STAT	65.8	29.8	43.9	21.1	0.9	0.	0.	4.4
VO AT	66.7	17.5	21.9	27.2	15.8	1.8	0.	15.8
DYN 1	62.3	14.0	30.7	24.6	7.0	0.	0.	23.7
ANL 2	58.8	24.6	37.7	14.0	6.1	0.9	0.	16.7
VKTRA	67.5	3.5	28.9	23.7	14.9	0.	0.	28.9
NUM A	60.5	19.3	31.6	23.7	5.3	0.	0.	20.2
KRFSK	59.6	7.9	21.1	31.6	7.0	0.	0.	32.5
DYN 2	58.8	23.7	36.8	21.1	0.9	0.	0.	17.5
KVNTM	42.1	9.6	13.2	14.0	14.0	0.9	0.	48.2
MT ST	42.1	11.4	20.2	13.2	8.8	0.	0.	46.5
TR EL2	31.6	16.7	26.3	5.3	0.	0.	0.	51.8
TR EL1	38.6	23.7	27.2	10.5	0.9	0.	0.	37.7
SLHT	50.9	21.9	25.4	21.9	3.5	0.	0.	27.2
FUNK	50.9	15.8	28.1	14.0	8.8	0.	0.	33.3

	GODK	UDRKD	BGT 3	BTG 4	BTG 5	BGT 6	BTG 7	EJTNT
LIN A	81	30	27	36	14	4	0	3
FSK I	84	19	46	28	10	0	0	11
ANL 1	87	20	53	21	12	1	0	7
STAT	75	34	50	24	1	0	0	5
VO AT	76	20	25	31	18	2	0	18
DYN 1	71	16	35	28	8	0	0	27
ANL 2	67	28	43	16	7	1	0	19
VKTRA	77	4	33	27	17	0	0	33
NUM A	69	22	36	27	6	0	0	23
KRFSK	68	9	24	36	8	0	0	37
DYN 2	67	27	42	24	1	0	0	20
KVNTM	48	11	15	16	16	1	0	55
MT ST	48	13	25	15	10	0	0	53
TR EL2	36	19	30	6	0	0	0	59
TR EL1	44	27	31	12	1	0	0	43
SLHT	58	25	29	25	4	0	0	31
FUNK	58	18	32	16	10	0	0	38

GODKÄNDA ENLIGT KVAL1 KVAL2 KVAL3

(RÄKNAT I PROCENT AV SAMTLIGA MED RESP. KVALITE)

LIN A	37.8	73.0	100.
FSK I	32.4	86.5	100.
ANL 1	48.6	81.1	97.5
STAT	21.6	75.7	97.5
VO AT	27.0	70.3	100.
DYN 1	21.6	64.9	97.5
ANL 2	29.7	48.6	95.
VKTRA	32.4	67.6	100.
NUM A	16.2	64.9	97.5
KRFSK	24.3	56.8	95.
DYN 2	29.7	45.9	97.5
KVNTM	10.8	29.7	82.5
MT ST	16.2	32.4	75.
TR EL2	5.4	37.8	50.
TR EL1	2.7	29.7	80.
SLHT	13.5	48.6	87.5
FUNK	13.5	48.6	87.5

	ANTAL PLUSSARE	LYCKADE PLUSSNINGAR
LIN A	16	0
FSK I	16	5
ANL 1	12	2
STAT	5	1
VO AT	11	3
DYN 1	8	3
ANL 2	4	0
VKTRA	9	3
NUM A	7	1
KRFSK	5	0
DYN 2	12	0
KVNTM	1	0
MT ST	2	0
TR EL2	1	0
TR EL1	1	0
SLHT	4	1
FUNK	2	1

ANTAL ELEVER 59
ANTAL BRA ELEVER = ANTAL DÅLIGA ELEVER = 55.3%
KVOT 4.50
ANTAL TENTAMENSTILLFÄLLEN I MEDEL PER ELEV 17.19
ANTAL ELEVER MED INGA GODKÄNDA TENTOR (I %) 0.

ANTAL KLARADE TENTOR	(I %)	ANTAL ELEVER	(I %)
(TENTORNA EJ VIKTADE EFTER IBORDES STORLEK)			
1	5.	1	1.69
2	10.	1	1.69
3	15.	1	1.69
4	20.	4	6.78
5	25.	5	8.47
6	30.	6	10.17
7	35.	1	1.69
8	40.	2	3.39
9	45.	2	3.39
10	50.	1	1.69
11	55.	8	13.56
12	60.	1	1.69
13	65.	5	8.47
14	70.	2	3.39
15	75.	3	5.08
16	80.	2	3.39
17	85.	4	6.78
18	90.	3	5.08
19	95.	7	11.86
20	100.	0	0.

ANTAL KLARADE TENTOR I % (TENTORNA VIKTADE EFTER INBORDES STORLEK)	ANTAL ELEVER	(I %)	(KUMULERAT ANTAL I %)	ELEVERNA FÖRDELADE PÅ KVALITÉ		
				KVALITÉ 1	2	3
				SUMMA 20	23	16
5.33	1	1.69	100.0	1	0	0
6.67	1	1.69	98.3	1	0	0
10.	0	0.	96.6	0	0	0
13.33	1	1.69	96.6	1	0	0
16.67	1	1.69	94.9	1	0	0
20.	4	6.78	93.2	4	0	0
23.33	5	8.47	86.4	5	0	0
26.67	2	5.39	78.0	2	0	0
30.	4	6.78	74.6	4	0	0
33.33	2	5.39	67.8	1	1	0
36.67	0	0.	64.4	0	0	0
40.	1	1.69	64.4	0	1	0
43.33	1	1.69	62.7	0	1	0
46.67	0	0.	61.0	0	0	0
50.	4	6.78	61.0	0	4	0
53.33	2	5.39	54.2	0	2	0
56.67	4	6.78	50.8	0	4	0
60.	1	1.69	44.1	0	1	0
63.33	2	5.39	42.4	0	2	0
66.67	4	6.78	39.0	0	4	0
70.	0	0.	32.2	0	0	0
73.33	2	5.39	32.2	0	2	0
76.67	1	1.69	28.8	0	1	0
80.	1	1.69	27.1	0	0	1
83.33	3	5.08	25.4	0	0	3
86.67	2	5.39	20.3	0	0	2
90.	2	5.39	16.9	0	0	2
93.33	7	11.86	13.6	0	0	7
96.67	1	1.69	1.7	0	0	1
100.	0	0.	0.	0	0	0

	GODKD	UDRKD	BTG 3	BTG 4	BTG 5	BTG 6	BTG 7	EJTNT
	(ANTAL ELEVER I PROCENT)							
FSK I	74.6	11.9	23.7	42.4	6.8	1.7	0.	13.6
HØLLF	50.8	37.3	11.9	25.4	10.2	3.4	0.	11.9
ANL 1	67.8	27.1	30.5	13.6	23.7	0.	0.	5.1
ANL 2	62.7	16.9	27.1	25.4	5.1	5.1	0.	20.3
LIN A	76.3	20.3	22.0	32.2	20.3	1.7	0.	3.4
VKTRA	62.7	5.1	16.9	30.5	15.6	1.7	0.	32.2
STAT	67.8	25.4	39.0	20.3	8.5	0.	0.	6.8
VØGLR	20.3	25.4	16.9	3.4	0.	0.	0.	54.2
DYN 1	61.0	32.2	28.8	16.9	15.3	0.	0.	6.8
NUM A	62.7	13.6	18.6	27.1	13.6	3.4	0.	23.7
RITTK	83.1	5.1	30.5	52.5	0.	0.	0.	11.9
MA FK	50.8	3.4	20.3	25.4	5.1	0.	0.	45.8
FUNK	45.8	8.5	15.3	16.9	13.6	0.	0.	45.8
KVNTM	52.5	3.4	13.6	20.3	18.6	0.	0.	44.1
SLHT	64.4	13.6	16.9	23.7	23.7	0.	0.	22.0
MT ST	35.6	1.7	10.2	8.5	16.9	0.	0.	62.7
DYN 2	66.1	8.5	5.1	40.7	20.3	0.	0.	25.4
TR EL1	37.3	18.6	18.6	13.6	5.1	0.	0.	44.1
TR EL2	33.9	30.5	23.7	5.1	5.1	0.	0.	35.6
MVTMS	40.7	0.	15.3	15.3	10.2	0.	0.	59.3

	GODK	UDRKD	BGT 3	BTG 4	BTG 5	BGT 6	BTG 7	EJTNT
FSK 1	44	7	14	25	4	1	0	8
HØLLF	30	22	7	15	6	2	0	7
ANL 1	40	16	18	8	14	0	0	3
ANL 2	37	10	16	15	3	3	0	12
LIN A	45	12	13	19	12	1	0	2
VKTRA	37	3	10	18	8	1	0	19
STAT	40	15	25	12	5	0	0	4
VØGLR	12	15	10	2	0	0	0	32
DYN 1	56	19	17	10	9	0	0	4
NUM A	37	8	11	16	8	2	0	14
PITTK	49	3	18	31	0	0	0	7
MA FK	30	2	12	15	3	0	0	27
FUNK	27	5	9	10	8	0	0	27
KVNTM	31	2	8	12	11	0	0	26
SLHT	38	8	10	14	14	0	0	13
MT ST	21	1	6	5	10	0	0	37
DYN 2	39	5	3	24	12	0	0	15
TR EL1	22	11	11	8	3	0	0	26
TR EL2	20	18	14	3	3	0	0	21
MVTMS	24	0	9	9	6	0	0	35

GODKÄNDA ENLIGT KVAL1 KVAL2 KVAL3

(RÄKNAT I PROCENT AV SAMTLIGA MED RESP. KVALITE)

FSK I	50.	78.3	100.
HÄLLF	5.	56.5	100.
ANL 1	30.	78.3	100.
ANL 2	30.	65.2	100.
LIN A	45.	87.0	100.
VKTRA	20.	87.0	81.2
STAT	40.	73.9	93.7
VÄGLR	0.	17.4	50.
DYN 1	40.	52.2	100.
NUM A	25.	69.6	100.
RITTK	60.	95.7	93.7
MA FK	10.	56.5	93.7
FUNK	15.	47.8	81.2
KVNTM	20.	52.2	93.7
SLHT	25.	82.6	87.5
MT ST	10.	26.1	81.2
DYN 2	35.	69.6	100.
TR EL1	10.	26.1	87.5
TR EL2	15.	30.4	62.5
MVTMS	5.	39.1	87.5

	ANTAL PLUSSARE	LYCKADE PLUSSNINGAR
FSK I	4	0
HÄLLF	1	0
ANL 1	7	3
ANL 2	5	0
LIN A	9	2
VKTRA	0	0
STAT	7	3
VÄGLR	0	0
DYN 1	2	1
NUM A	5	2
RITTK	5	0
MA FK	2	0
FUNK	1	0
KVNTM	1	0
SLHT	7	1
MT ST	0	0
DYN 2	1	0
TR EL1	0	0
TR EL2	0	0
MVTMS	0	0

```
RUNID SIM    ACCT 208461        PROJ SIM           PRIO M&L

START 15:36:12 SEP 26,1973      FIN 15:38:151SEP 26,1973    QUEUED 00:00

IMAGES READ 4147         PAGES 36        IMAGES PUNCHED 0

I/O BY GROUP:        DRUM       DISC
MILLISECONDS:       16565      15696

DELKOSTN.:                    15.37 CPU;     7.45 KARNMODUL;     0.00 REALTID
   0.00 BANDMONT+PLOTT+FARGB+REMSOR;        0.00 UPPKOPPL;      0.00 TEMP FIL
   0.00 KORT UT;          34.18 KORT IN;     6.00 SIDOR

  PRIS KR    TIME: TOTAL:00:02:11.230  CPU:00:00:36.451  MEM:00:00:33.527
    63.00          CC/FR:00:00:28.989  I/O:00:00:32.262 WAIT:00:00:00.576
```

Appendix 3: Styrkort till SIMSCRIPT-program

(RUN- och FIN-kort utelämnas på listorna)
Bandprogram fordrar körorder och är relativt dyra att exekvera.
De används om man vill köra ett program bara några få gånger.
(␣ = mellanslag)

```
Styrkort
" ASG,T␣11
" ASG,TF␣2373,C9,2373
" ASG,T␣SIM*LIB
" FIND,A␣2373. SIM
" COPIN,A␣2373. SIM, SIM*ABS. SIM
" SIM*ABS.SIM,S
  <program>

" FIND,S␣2373. SIMREG
" COPIN,S␣2373. SIMREG,TPFØ.SIMREG
" ADD,P␣11.
" MOVE␣2373.,1
" COPIN␣2373., SIM*LIB.
" PREP␣SIM*LIB.
" MAP,IS␣SIMTEST
" LIB␣SIM*LIB.
EQU␣B11/11
"XQT␣SIMTEST
   <data>
```

Om man vill använda ett program många gånger lägger man ut det på en fil. (Här har filen namnet F2). Bandprogrammet körs först en gång med
"ASG,TF␣2373,C9,2373 utbytt mot "ASG,UP␣SIM*ABS., F2 och
"ASG,T␣SIM* LIB ändras till "ASG,UP␣SIM* LIB., F2.
Efter "ADD,P␣11. läggs följande kort in:
"COPY,S␣TPFØ.SIMREG,SIM*ABS.SIMREG.
Därefter kan programmet köras utan körorder med följande styrkort (programbeteckning är TENTOR):

```
"ASG,T␣11.
"ASG,AX␣SIM*ABS.
"ASG,AX␣SIM*LIB.
"SIM*ABS.SIM,S
   <program>
```

```
"COPY,S␣SIM*ABS. SIMREG,TPFØ.
"ADD,P␣11.
"MAP,IS␣TENTOR
 LIB␣SIM*LIB
 EQU␣B11/11
"XQT␣TENTOR
  <data>
```

Lunds datacentral har en självbetjäning, där det går fortare att få ut resultat än genom kundmottagningen. Programmen får emellertid inte överskrida en maxtid och ett maxsidantal. Man kan då dela sitt program i två delar, där den ena innehåller själva programmet och den andra datakorten. Ovanstående kort ändras då enligt följande:

```
   Program 1
"MAP,IS␣TENTOR byts mot två kort nämligen:
"ASG,AX␣T*FIL.
"MAP,IS␣IN,T*FIL. TENTOR
"XQT␣TENTOR tas bort och programmet slutar med kortet
EQU␣B11/11.

   Program 2
"ASG,AX␣T*FIL.
"XQT␣T*FIL. TENTOR
  <data>
```

Första gången det delade programmet körs läggs följande kort in direkt efter run-kortet i program 1:

```
"ASG,UP␣T*FIL.,F2
"FREE␣T*FIL.
```

Dessa kort tas sedan bort.
Flera program kan finnas på samma fil. Man ändrar då bara programbeteckning t. ex. i det enkla tillämpningsexemplet byts TENTOR ut mot MASKIN på programmets styrkort. Får programmet för många sidor för självbetjäning och man inte behöver någon utskrift av programdelen kan man ändra "SIM*ABS.SIM,S till "SIM*ABS.SIM,N (för både delat och odelat program). Har man glömt, vad som ligger på filen, använder man styrkortet "PRT,T␣T*FIL. och får då utskrift av filens innehåll.
Det man lagt ut på en fil ligger kvar och ändras ej.
När man därför inte behöver den mer, måste man förstöra filen. Detta görs med DELETE␣FIL*F2.