```
BLOCK
(* BANK DEPARTMENT SERVICE SIMULATION *)



UNIT PRIORITYQUEUE: CLASS;
  (* HEAP AS BINARY LINKED TREE WITH FATHER LINK*)



    UNIT QUEUEHEAD: CLASS;
      (* HEAP ACCESING MODULE *)
         VAR LAST,ROOT:NODE;

         UNIT MIN: FUNCTION: ELEM;
             BEGIN
           IF ROOT=/= NONE THEN RESULT:=ROOT.EL FI;
            END MIN;

         UNIT INSERT: PROCEDURE(R:ELEM);
           (* INSERTION INTO HEAP *)
              VAR X,Z:NODE;
            BEGIN
                X:= R.LAB;
                IF LAST=NONE THEN
                  ROOT:=X;
                  ROOT.LEFT,ROOT.RIGHT,LAST:=ROOT
                ELSE
                  IF LAST.NS=0 THEN
                    LAST.NS:=1;
                    Z:=LAST.LEFT;
                    LAST.LEFT:=X;
                    X.UP:=LAST;
```

```
                    X.LEFT:=Z;
                    Z.RIGHT:=X;
                  ELSE
                    LAST.NS:=2;
                    Z:=LAST.RIGHT;
                    LAST.RIGHT:=X;
                    X.RIGHT:=Z;
                    X.UP:=LAST;
                    Z.LEFT:=X;
                    LAST.LEFT.RIGHT:=X;
                    X.LEFT:=LAST.LEFT;
                    LAST:=Z;
                   FI
                 FI;
                 CALL CORRECT(R,FALSE)
                 END INSERT;

UNIT DELETE: PROCEDURE(R: ELEM);
    VAR X,Y,Z:NODE;
    BEGIN
    X:=R.LAB;
    Z:=LAST.LEFT;
    IF LAST.NS =0 THEN
        Y:= Z.UP;
        Y.RIGHT:= LAST;
        LAST.LEFT:=Y;
        LAST:=Y;
              ELSE
        Y:= Z.LEFT;
        Y.RIGHT:= LAST;
         LAST.LEFT:= Y;
               FI;
     Z.EL.LAB:=X;
```

```
      X.EL:= Z.EL;
      LAST.NS:= LAST.NS-1;
      R.LAB:=Z;
      Z.EL:=R;
      IF X.LESS(X.UP) THEN CALL CORRECT(X.EL,FALSE)
                    ELSE CALL CORRECT(X.EL,TRUE) FI;
    END DELETE;

UNIT CORRECT: PROCEDURE(R:ELEM,DOWN:BOOLEAN);
  (* CORRECTION OF THE HEAP WITH STRUCTURE BROKEN
BY R *)
    VAR X,Z:NODE,T:ELEM,FIN,LOG:BOOLEAN;
    BEGIN
    Z:=R.LAB;
    IF DOWN THEN
        WHILE NOT FIN DO
             IF Z.NS =0 THEN FIN:=TRUE ELSE
                IF Z.NS=1 THEN X:=Z.LEFT ELSE
                IF Z.LEFT.LESS(Z.RIGHT) THEN X:=Z.LEFT ELSE
X:=Z.RIGHT
                 FI; FI;
                IF Z.LESS(X) THEN FIN:=TRUE ELSE
                    T:=X.EL;
                    X.EL:=Z.EL;
                    Z.EL:=T;
                    Z.EL.LAB:=Z;
                   X.EL.LAB:=X
                FI; FI;
            Z:=X;
                 OD
          ELSE
   X:=Z.UP;
   IF X=NONE THEN LOG:=TRUE ELSE LOG:=X.LESS(Z); FI;
```

```
    WHILE NOT LOG DO
        T:=Z.EL;
        Z.EL:=X.EL;
         X.EL:=T;
        X.EL.LAB:=X;
        Z.EL.LAB:=Z;
        Z:=X;
        X:=Z.UP;
         IF X=NONE THEN LOG:=TRUE ELSE LOG:=X.LESS(Z);
          FI;
             OD
    FI;
 END CORRECT;

END QUEUEHEAD;


UNIT NODE: CLASS (EL:ELEM);
  (* ELEMENT OF THE HEAP *)
     VAR LEFT,RIGHT,UP: NODE, NS:INTEGER;
     UNIT LESS: FUNCTION(X:NODE): BOOLEAN;
        BEGIN
        IF X= NONE THEN RESULT:=FALSE
                ELSE RESULT:=EL.LESS(X.EL) FI;
        END LESS;
    END NODE;


UNIT ELEM: CLASS(PRIOR:REAL);
  (* PREFIX OF INFORMATION TO BE STORED IN NODE *)
   VAR LAB: NODE;
   UNIT VIRTUAL LESS: FUNCTION(X:ELEM):BOOLEAN;
          BEGIN
```

```
        IF X=NONE THEN RESULT:= FALSE ELSE
                RESULT:= PRIOR< X.PRIOR FI;
        END LESS;
    BEGIN
    LAB:= NEW NODE(THIS ELEM);
    END ELEM;


END PRIORITYQUEUE;



UNIT SIMULATION: PRIORITYQUEUE CLASS;
(* THE LANGUAGE FOR SIMULATION PURPOSES *)

  VAR CURR: SIMPROCESS,  (*ACTIVE PROCESS *)
     PQ:QUEUEHEAD,  (* THE TIME AXIS *)
      MAINPR: MAINPROGRAM;


     UNIT SIMPROCESS: COROUTINE;
       (* USER PROCESS PREFIX *)
          VAR EVENT,  (* ACTIVATION MOMENT NOTICE *)
             EVENTAUX: EVENTNOTICE,
             (* THIS IS FOR AVOIDING MANY NEW CALLS AS AN
RESULT OF *)
             (* SUBSEQUENT PASSIVATIONS AND ACTIVATIONS
*)
             FINISH: BOOLEAN;

          UNIT IDLE: FUNCTION: BOOLEAN;
               BEGIN
               RESULT:= EVENT= NONE;
```

```
        END IDLE;

    UNIT TERMINATED: FUNCTION :BOOLEAN;
        BEGIN
        RESULT:= FINISH;
        END TERMINATED;

    UNIT EVTIME: FUNCTION: REAL;
    (* TIME OF ACTIVATION *)
        BEGIN
        IF IDLE THEN CALL ERROR1;
                    FI;
        RESULT:= EVENT.EVENTTIME;
        END EVTIME;

  UNIT ERROR1:PROCEDURE;
        BEGIN
        ATTACH(MAIN);
        WRITELN(" AN ATTEMPT TO ACCESS AN IDLE
PROCESS TIME");
        END ERROR1;

   UNIT ERROR2:PROCEDURE;
        BEGIN
        ATTACH(MAIN);
        WRITELN(" AN ATTEMPT TO ACCESS A TERMINATED
PROCESS TIME");
        END ERROR2;
    BEGIN

    RETURN;
    INNER;
    FINISH:=TRUE;
```

```
            CALL PASSIVATE;
           CALL ERROR2;
         END SIMPROCESS;


UNIT EVENTNOTICE: ELEM CLASS;
  (* A PROCESS ACTIVATION NOTICE TO BE PLACED ONTO
THE TIME AXIS PQ *)
     VAR EVENTTIME: REAL, PROC: SIMPROCESS;

     UNIT VIRTUAL LESS: FUNCTION(X:
EVENTNOTICE):BOOLEAN;
      (* OVERWRITE THE FORMER VERSION CONSIDERING
EVENTTIME *)
               BEGIN
               IF X=NONE THEN RESULT:= FALSE ELSE
               RESULT:= EVENTTIME< X.EVENTTIME OR
               (EVENTTIME=X.EVENTTIME AND PRIOR< X.PRIOR);
FI;

             END LESS;
   END EVENTNOTICE;


UNIT MAINPROGRAM: SIMPROCESS CLASS;
 (* IMPLEMENTING MASTER PROGRAM AS A PROCESS *)
     BEGIN
     DO ATTACH(MAIN) OD;
     END MAINPROGRAM;

UNIT TIME:FUNCTION:REAL;
 (* CURRENT VALUE OF SIMULATION TIME *)
    BEGIN
```

```
    RESULT:=CURRENT.EVTIME
    END TIME;

UNIT CURRENT: FUNCTION: SIMPROCESS;
  (* THE FIRST PROCESS ON THE TIME AXIS *)
    BEGIN
    RESULT:=CURR;
    END CURRENT;

UNIT SCHEDULE: PROCEDURE(P:SIMPROCESS,T:REAL);
 (* ACTIVATION OF PROCESS P AT TIME T AND DEFINITION
OF "PRIOR"- PRIORITY *)
 (* WITHIN TIME MOMENT T                                    *)
     BEGIN
     IF T<TIME THEN T:= TIME FI;
     IF P=CURRENT THEN CALL HOLD(T-TIME) ELSE
     IF P.IDLE AND P.EVENTAUX=NONE THEN (* HAS NOT
BEEN SCHEDULED YET*)
             P.EVENT,P.EVENTAUX:= NEW
EVENTNOTICE(RANDOM);
             P.EVENT.PROC:= P;
                              ELSE
      IF P.IDLE (* P HAS ALREADY BEEN SCHEDULED *) THEN
            P.EVENT:= P.EVENTAUX;
            P.EVENT.PRIOR:=RANDOM;
                                 ELSE
  (* NEW SCHEDULING *)
            P.EVENT.PRIOR:=RANDOM;
            CALL PQ.DELETE(P.EVENT)
                          FI; FI;
    P.EVENT.EVENTTIME:= T;
    CALL PQ.INSERT(P.EVENT) FI;
END SCHEDULE;
```

```
UNIT HOLD:PROCEDURE(T:REAL);
 (* MOVE THE ACTIVE PROCESS T MINUTES BACK ALONG PQ
*)
 (* REDEFINE PRIOR                          *)
    BEGIN
    CALL PQ.DELETE(CURRENT.EVENT);
    CURRENT.EVENT.PRIOR:=RANDOM;
    IF T<0 THEN T:=0; FI;
     CURRENT.EVENT.EVENTTIME:=TIME+T;
    CALL PQ.INSERT(CURRENT.EVENT);
    CALL CHOICEPROCESS;
    END HOLD;

UNIT PASSIVATE: PROCEDURE;
  (* REMOVE THE ACTVE PROCESS FROM PQ AND ACTIVATE
THE NEXT ONE *)
    BEGIN
     CALL PQ.DELETE(CURRENT.EVENT);
     CURRENT.EVENT:=NONE;
     CALL CHOICEPROCESS
    END PASSIVATE;

UNIT RUN: PROCEDURE(P:SIMPROCESS);
 (* ACTIVATE P IMMEDIATELY AND DELAY THE FORMER FIRST
PROCESS BY REDEFINING*)
 (* PRIOR                                    *)
    BEGIN
    CURRENT.EVENT.PRIOR:=RANDOM;
    IF NOT P.IDLE THEN
          P.EVENT.PRIOR:=0;
          P.EVENT.EVENTTIME:=TIME;
          CALL PQ.CORRECT(P.EVENT,FALSE)
```

```
                ELSE
     IF P.EVENTAUX=NONE THEN
          P.EVENT,P.EVENTAUX:=NEW EVENTNOTICE(0);
          P.EVENT.EVENTTIME:=TIME;
          P.EVENT.PROC:=P;
          CALL PQ.INSERT(P.EVENT)
                   ELSE
           P.EVENT:=P.EVENTAUX;
           P.EVENT.PRIOR:=0;
           P.EVENT.EVENTTIME:=TIME;
           P.EVENT.PROC:=P;
           CALL PQ.INSERT(P.EVENT);
                    FI;FI;
     CALL CHOICEPROCESS;
END RUN;

UNIT CANCEL:PROCEDURE(P: SIMPROCESS);
 (* REMOVE PROCESS P FROM PQ AND CONTINUE
SIMULATION *)
   BEGIN
   IF P= CURRENT THEN CALL PASSIVATE ELSE
    CALL PQ.DELETE(P.EVENT);
    P.EVENT:=NONE;  FI;
 END CANCEL;

UNIT CHOICEPROCESS:PROCEDURE;
 (* CHOOSE THE FIRST PROCESS FROM PQ TO BE ACTIVATED
*)
   VAR P:SIMPROCESS;
   BEGIN
   P:=CURR;
   CURR:= PQ.MIN QUA EVENTNOTICE.PROC;
    IF CURR=NONE THEN WRITE(" ERROR IN THE HEAP");
```

```
WRITELN;
                ATTACH(MAIN);
            ELSE ATTACH(CURR); FI;
END CHOICEPROCESS;

BEGIN
  PQ:=NEW QUEUEHEAD;  (* SIMULATION TIME AXIS*)
  CURR,MAINPR:=NEW MAINPROGRAM;
  MAINPR.EVENT,MAINPR.EVENTAUX:=NEW EVENTNOTICE(0);
  MAINPR.EVENT.EVENTTIME:=0;
  MAINPR.EVENT.PROC:=MAINPR;
  CALL PQ.INSERT(MAINPR.EVENT);
  (* THE FIRST PROCESS TO BE ACTIVATED IS MAIN PROGRAM
*)
  INNER;
  PQ:=NONE;
END SIMULATION;



UNIT LISTS:SIMULATION CLASS;
 (* WE WISH TO USE LISTS FOR QUEUEING PROCESSES
DURING SIMULATION*)

        UNIT LINKAGE:CLASS;
         (*WE WILL USE TWO WAY LISTS *)
            VAR SUC1,PRED1:LINKAGE;
                  END LINKAGE;
         UNIT HEAD:LINKAGE CLASS;
         (* EACH LIST WILL HAVE ONE ELEMENT ESTABLISHED
*)
                UNIT FIRST:FUNCTION:LINK;
                      BEGIN
```

```
            IF SUC1 IN LINK THEN RESULT:=SUC1
                        ELSE RESULT:=NONE FI;
                END;
        UNIT EMPTY:FUNCTION:BOOLEAN;
                BEGIN
                RESULT:=SUC1=THIS LINKAGE;
                END EMPTY;
      BEGIN
      SUC1,PRED1:=THIS LINKAGE;
       END HEAD;

UNIT LINK:LINKAGE CLASS;
 (* ORDINARY LIST ELEMENT PREFIX *)
        UNIT OUT:PROCEDURE;
              BEGIN
              IF SUC1=/=NONE THEN
                  SUC1.PRED1:=PRED1;
                  PRED1.SUC1:=SUC1;
                  SUC1,PRED1:=NONE FI;
               END OUT;
        UNIT INTO:PROCEDURE(S:HEAD);
              BEGIN

              CALL OUT;
              IF S=/= NONE THEN
                  IF S.SUC1=/=NONE THEN
                        SUC1:=S;
                        PRED1:=S.PRED1;
                        PRED1.SUC1:=THIS LINKAGE;
                        S.PRED1:=THIS LINKAGE;
                            FI FI;
                END INTO;
      END LINK;
```

```
    UNIT ELEM:LINK CLASS(SPROCESS:SIMPROCESS);
    (* USER DEFINED  PROCESS WILL BE JOINED INTO LISTS
*)
                END ELEM;

   END LISTS;




  (*BEGIN OF BANK DEPARTMENT SIMULATION*)


  UNIT OFFICE:LISTS CLASS; (*AN OFFICE*)

    UNIT TILL:SIMPROCESS CLASS(QUEUE:HEAD);
    (* TELLER WITH CUSTOMERS QUEUEING UP *)
          UNIT VIRTUAL SERVICE:PROCEDURE;
           (* SERVICE OF THIS TELLER WILL BE PRECISED LATER
*)
                          END SERVICE;
            VAR CSTM:CUSTOMER,  (*THE CUSTOMER BEING
SERVED*)
                REST,PAUSE:REAL;

            BEGIN
            PAUSE:=TIME;
            DO
            REST:=REST+TIME-PAUSE;
            WHILE NOT QUEUE.EMPTY DO
             (* SERVE ALL QUEUE *)
```

```
                CSTM:=QUEUE.FIRST QUA ELEM.SPROCESS;
                CALL SERVICE;
                CALL SCHEDULE(CSTM,TIME);
                        OD;
        PAUSE:=TIME;
        CALL PASSIVATE
        OD;
    END TILL;

  UNIT CUSTOMER:SIMPROCESS CLASS;

        VAR ELLIST:ELEM, K:INTEGER;
        UNIT ARRIVAL:PROCEDURE(S:TILL);
        (* ATTACHING TELLER S *)
            BEGIN
            IF S=/=NONE THEN
              ELLIST:=NEW ELEM(THIS CUSTOMER);
              CALL ELLIST.INTO(S.QUEUE);
              IF S.IDLE THEN CALL SCHEDULE(S,TIME) FI;
              CALL PASSIVATE; FI;
            END ARRIVAL;
     END CUSTOMER;

 END OFFICE;



UNIT BANKDEPARTMENT:OFFICE CLASS;


    UNIT COUNTER:TILL CLASS;
        VAR PAYTIME:REAL; (*RANDOM SERVICE TIME*)
        UNIT VIRTUAL SERVICE:PROCEDURE;
```

```
            BEGIN
            WRITELN(" THE PAY DESK  SERVES CUSTOMER
NO",CSTM.K,
                  " AT",TIME:10:4);
            CALL CSTM.ELLIST.OUT;
            PAYTIME:=RANDOM*2+2;
            CALL HOLD(PAYTIME);
            END SERVICE;
   END COUNTER;


   UNIT TELLER:TILL CLASS(NUMBER:INTEGER);
         VAR SERVICETIME:REAL;
         UNIT VIRTUAL SERVICE:PROCEDURE;
           VAR N:INTEGER;
           BEGIN
           WRITELN(" THE TELLER NO",NUMBER," WAS IDLE
FOR",REST:10:4,
                  " SEC");
            CALL CSTM.ELLIST.OUT;
            N:=CSTM QUA BANKCUSTOMER.NO;
            WRITELN(" THE CUSTOMER NO",CSTM.K,
                  " BEGINS TO BE SERVED BY THE TELLER
NO",NUMBER,
                  " AT",TIME:10:4);
            ACCOUNT(N):=ACCOUNT(N)+CSTM QUA
BANKCUSTOMER.AMOUNT;
            IF ACCOUNT(N)<0 THEN CALL
CSTM.ARRIVAL(CONTROL);FI;
            SERVICETIME:=RANDOM*7+3;
            CALL HOLD(SERVICETIME);

            END SERVICE;
```

```
        END TELLER;


   UNIT BANKCUSTOMER:CUSTOMER
CLASS(NO:INTEGER,AMOUNT:REAL);
   (* BANK CUSTOMER. AMOUNT- THE MONEY TO BE PAID AT
THE BANK *)
          VAR
ARRIVALTIME,STAYTIME:REAL,CHOOSETELLER:INTEGER;
            BEGIN
            I:=I+1;
            K:=I;
            ARRIVALTIME:=TIME;
            WRITELN(" THE CUSTOMER NO",K," ARRIVED
AT",TIME:10:4);
            CHOOSETELLER:=RANDOM*5+1;
            CALL ARRIVAL(TELLERS(CHOOSETELLER));
            IF AMOUNT<0 THEN CALL ARRIVAL(CTR); FI;
            STAYTIME:=TIME-ARRIVALTIME;
            WRITELN(" THE CUSTOMER NO",K," STAYED AT THE
BANK FOR",
                  STAYTIME:10:4," SEC; STATE OF
ACCOUNT",ACCOUNT(NO):10:4);
          END BANKCUSTOMER;

  VAR TELLERS:ARRAYOF TELLER,ACCOUNT:ARRAYOF REAL;
  VAR CTR:COUNTER, CONTROL:TILL,I:INTEGER;

    BEGIN   (* NEW BANK DEPARTMENT GENERATION *)
   CTR:=NEW COUNTER(NEW HEAD);
   ARRAY TELLERS DIM(1:5);  (* WE DEAL WITH 5 TELLES *)
   FOR I:=1 TO 5 DO  TELLERS(I):=NEW TELLER(NEW HEAD,I);
OD;
```

```
   ARRAY ACCOUNT DIM(1:100);
   (* WE DEAL WITH 100 ACOUNTS IN THIS BANK
DEPARTMENT *)
   FOR I:=1 TO 100 DO  ACCOUNT(I):=RANDOM*901+100; OD;
              (* AN ACCOUNT VALUE CAN FLUCTUATE FROM 100
TO 1000$ *)
   I:=0;
 END BANKDEPARTMENT;



 BEGIN (* OF PROGRAM *)
  PREF BANKDEPARTMENT BLOCK
      UNIT GENERATOR:SIMPROCESS CLASS;
       (* CUSTOMERS GENERATION *)
          BEGIN
          DO
          CALL SCHEDULE(NEW
BANKCUSTOMER(RANDOM*100+1,
                     RANDOM*9996+5),TIME);
          CALL HOLD(RANDOM*10);
          CALL SCHEDULE(NEW
BANKCUSTOMER(RANDOM*100+1,
                   -(RANDOM*900+5)),TIME);
          CALL HOLD(RANDOM*10);
          OD
          END GENERATOR;
     BEGIN
     WRITELN(" BANK DEPARTMENT SERVICE SIMULATION");
     WRITELN;
     CALL SCHEDULE(NEW GENERATOR,TIME);
     CALL HOLD (40);
      END
```

END