1.

a. Här är ett sätt att skriva `IF`-satsen m h a `CASE` (det finns fler sätt).

```
CASE villkor OF
  TRUE:  satsföljd1
| FALSE: satsföljd2
END
```

b. Typ 1: syntaxfel, typfel, ...

Typ 2: division med noll, indexering utanför gränserna, ...

Typ 3: programmet beräknar fel resultat, t ex p g a att man har skrivit `A+B` när man menade `A-B`.

2.

a. Följande rad skrivs ut: `3 2 1 0 0 1 2 3`

b. Följande rad skrivs ut: `3 2 1 0 0 0 0 0`

3.

a. Följande antas vara importerat i modulen:

```
FROM Strings IMPORT Length;
```

Funktionen får heta `Passar`.

```
PROCEDURE Passar(VAR mnster,ord:ARRAY OF CHAR):BOOLEAN;
VAR i:CARDINAL;
BEGIN
  IF Length(mnster)#Length(ord) THEN
    RETURN FALSE
  ELSE
    FOR i:=0 TO Length(ord)-1 DO
      IF (mnster[i]#"?") AND (mnster[i]#ord[i]) THEN
        RETURN FALSE
      END
    END;
    RETURN TRUE
  END
END Passar;
```

b. Proceduren får heta `PassandeOrd`.

```
PROCEDURE PassandeOrd(mnster:ARRAY OF CHAR; VAR ord:ARRAY OF Ord; VAR
antal:CARDINAL);
VAR l:Ordlista;
BEGIN
  antal:=0;
  l:=ordlista;
  WHILE l#NIL DO
    IF Passar(mnster,l^.ord) THEN
      IF antal<=HIGH(ord) THEN
        ord[antal]:=l^.ord
      END;
      INC(antal)
    END;
    l:=l^.resten
  END
END PassandeOrd;
```


4.

Lösningarna är gjorda för Mocka-systemet. Följande antas vara importerat i modulen:

```
FROM Strings IMPORT compare;
FROM InOut IMPORT WriteCard,WriteLn;
```

a. Procedurerna `Init` och `WriteList`:

```
PROCEDURE Init(VAR l:NumberList);
BEGIN
  l.count:=0
END Init;

PROCEDURE WriteList(VAR l:NumberList);
VAR i:CARDINAL;
BEGIN
  FOR i:=1 TO l.count DO
    WriteCard(l.number[i],1); WriteLn
  END
END WriteList;
```

b. Procedurerna `Insert` och `Delete`:

```
PROCEDURE Insert(n:Number; VAR l:NumberList);
VAR i:CARDINAL;
BEGIN
  i:=l.count;
  WHILE (i>0) AND (n<l.number[i]) DO
    l.number[i+1]:=l.number[i];
    DEC(i)
  END;
  l.number[i+1]:=n;
  INC(l.count)
END Insert;

PROCEDURE Find(n:Number; VAR l:NumberList):CARDINAL; (* 0 = finns ej *)
VAR i:CARDINAL;
BEGIN
  FOR i:=1 TO l.count DO
    IF n=l.number[i] THEN
      RETURN i
    END
  END;
  RETURN 0
END Find; (* Find ingår inte i uppgiften, men används i Delete *)

PROCEDURE Delete(n:Number; VAR l:NumberList);
VAR i:CARDINAL;
BEGIN
  i:=Find(n,l);
  IF i>0 THEN
    WHILE i<l.count DO
      l.number[i]:=l.number[i+1];
      INC(i)
    END;
    DEC(l.count)
  END
END Delete;
```

5.

a. Typer för namnlistor representerade som binära sökträd:

```
TYPE Number = [100000..999999];

     NumberTree = POINTER TO NumberNode;
     NumberNode = RECORD
                    number: Number;
                    left, right: NumberTree
                  END;

     NumberList = NumberTree;
```

b. Följande antas vara importerat i modulen:

```
FROM Strings IMPORT compare;
FROM Storage IMPORT ALLOCATE;
FROM InOut IMPORT WriteCard,WriteLn;
```

Procedurerna `Init`, `WriteList` och `Insert`:

```
PROCEDURE Init(VAR l:NumberList);
BEGIN
  l:=NIL
END Init;

PROCEDURE NewLeaf(n:Number):NumberTree;
VAR t:NumberTree;
BEGIN
  NEW(t);
  t^.number:=n;
  Init(t^.left);
  Init(t^.right);
  RETURN t;
END NewLeaf;

PROCEDURE Insert(n:Number; VAR l:NumberList);
BEGIN
  IF l=NIL THEN
    l:=NewLeaf(n)
  ELSE
    IF n<l^.number THEN
      Insert(n,l^.left)
    ELSIF n>l^.number THEN
      Insert(n,l^.right)
    (* ELSE numret finns redan i listan *)
    END
  END
END Insert;

PROCEDURE WriteList(VAR l:NumberList);
BEGIN
  IF l#NIL THEN
    WriteList(l^.left);
    WriteCard(l^.number,1); WriteLn;
    WriteList(l^.right)
  END
END WriteList;
```


6.

a. Definitionsmodulen:

```
DEFINITION MODULE TextTerminal;

TYPE Style = (Normal, Bold, Underline, Blinking);

PROCEDURE Clear;
(* suddar hela skärmen *)

PROCEDURE MoveTo(line,column:CARDINAL);
(* flyttar markören till den angivna positionen *)

PROCEDURE ChangeStyle(style:Style);
(* byter till den angivna teckenstilen *)

END TextTerminal.
```

b. Implementeringsmodulen:

```
IMPLEMENTATION MODULE TextTerminal;

FROM InOut IMPORT Write,WriteString,WriteCard;

CONST Escape = CHR(27);

PROCEDURE CSI;
(* skriver ut startkoden (CSI=Control Sequence Introducer) *)
BEGIN
  Write(Escape); Write("[")
END CSI;

PROCEDURE Clear;
(* suddar hela skärmen *)
BEGIN
  CSI; WriteString("2J");
END Clear;

PROCEDURE MoveTo(line,column:CARDINAL);
(* flyttar markören till den angivna positionen *)
BEGIN
  CSI; WriteCard(line,1); Write(";"); WriteCard(column,1); Write("H")
END MoveTo;

PROCEDURE ChangeStyle(style:Style);
(* byter till den angivna teckenstilen *)

  PROCEDURE StyleCode(style:Style):CHAR;
  BEGIN
    CASE style OF
      Normal: RETURN "0"
    | Bold: RETURN "1"
    | Underline: RETURN "4"
    | Blinking: RETURN "5"
    END
  END StyleCode;

BEGIN
  CSI; Write(StyleCode(style)); Write("m")
END ChangeStyle;

END TextTerminal.
```