1.

	Sökning O(n)	Sökning O(log n)	Sökning O(1)
Insättning O(n)		c	
Insättning O(log n)		d	
Insättning O(1)	a, b		

2.

```
PROCEDURE IsPalindrome(s:ARRAY OF CHAR):BOOLEAN;
  VAR i,j:CARDINAL;

  PROCEDURE SkipChar(c:CHAR):BOOLEAN; (* Är c ett blank- eller skiljetecken *)
  BEGIN
    RETURN c<"A" (* fusklösning, eftersom siffror räknas som skiljetecken *)
  END SkipChar;
BEGIN
  i:=0; j:=Length(s)-1;
  WHILE i<j DO
    WHILE SkipChar(s[i]) AND (i<j) DO INC(i) END;
    WHILE SkipChar(s[j]) AND (j>i) DO DEC(j) END;
    IF CAP(s[i])#CAP(s[j]) THEN RETURN FALSE END;
    INC(i); DEC(j)
  END;
  RETURN TRUE;
END IsPalindrome;
```

3.

a.

```
TYPE Athlete = RECORD
                 name: ARRAY [1..30] OF CHAR;
                 country: ARRAY [1..3] OF CHAR;
                 birth: RECORD
                          date:Date;
                          place:ARRAY [1..20] OF CHAR
                        END;
                 height,weight: REAL;
                 shoes: ARRAY [1..20] OF CHAR;
                 married: BOOLEAN
               END;
```

b.

```
(* FROM InOut IMPORT WriteString,WriteReal,WriteLn; *)

PROCEDURE WriteAthlete(a:Athlete);
BEGIN
  WriteString("Country: "); WriteString(a.country); WriteLn;
  WriteString("Name    : "); WriteString(a.name); WriteLn;
  WriteString("Birth   : ");
    WriteDate(a.birth.date); WriteString(" ");
    WriteString(a.birth.place); WriteLn;
  WriteString("Height  : "); WriteReal(a.height,5,2); WriteString(" m"); WriteLn;
  WriteString("Weight  : "); WriteReal(a.weight,5,1); WriteString(" kg"); WriteLn;
  WriteString("Shoes   : "); WriteString(a.shoes); WriteLn;
```

```
  IF a.married THEN WriteString("not ") END;
  WriteString("married"); WriteLn
END WriteAthlete;
```

4.

a.

```
DEFINITION MODULE Tape;
FROM Movie IMPORT Movie,Minutes;

TYPE RecordingList;

     Tape = RECORD
              length:Minutes;
              recordings:RecordingList
            END;

PROCEDURE RecordNewMovie(VAR tape:Tape; m:Movie;
                         VAR ok:BOOLEAN; (* blir TRUE om filmen får plats *)
                         VAR where:Minutes);

(* + fler användbara procedurer *)

END Tape.
```

b.

```
IMPLEMENTATION MODULE Tape;
FROM Storage IMPORT ALLOCATE;
FROM Movie IMPORT Movie,Minutes,MovieLength;

TYPE RecordingList = POINTER TO Recording;
     Recording = RECORD
                   position: Minutes;   (* var på bandet inspelningen börjar *)
                   movie: Movie;        (* vilken film som är inspelad *)
                   next: RecordingList (* nästa film *)
                                        (* Listan ska hållas sorterad. *)
                 END;

PROCEDURE NewRecording(pos:Minutes; m:Movie; next:RecordingList):RecordingList;
  VAR new:RecordingList;
BEGIN
  NEW(new);
  new^.position:=pos;
  new^.movie:=m;
  new^.next:=next;
  RETURN new
END NewRecording;

PROCEDURE RecordNewMovie(VAR tape:Tape; movie:Movie;
                         VAR ok:BOOLEAN; VAR where:Minutes);

  PROCEDURE findFreeSpace(pos:Minutes; VAR r:RecordingList);
    (* pos = en position på bandet,
     * r = lista med inspelningar som börjar efter positionen pos *)
    VAR freeSpace:Minutes;
  BEGIN
    (* Räkna först ut hur mycket ledig plats det finns från pos
     * till början på nästa inspelning (eller slutet på banded) *)
```

```
    IF r=NIL THEN freeSpace:=tape.length-pos
             ELSE freeSpace:=r^.position-pos END;
    IF freeSpace>=MovieLength(movie) THEN
      (* Filmen får plats på position pos. *)
      ok:=TRUE;
      where:=pos;
      (* Stoppa in filmen på rätt plats i listan. *)
      r:=NewRecording(pos,movie,r)
    ELSIF r#NIL THEN
      (* Filmen får inte plats på positionen pos, men det finns fler
       * inspelningar på bandet.
       * Fortsätt med att prova vid slutet av nästa inspelning.
       *)
      findFreeSpace(pos+MovieLength(r^.movie),r^.next)
    ELSE
      (* Filmen får inte plats på positionen pos och det finns inga fler
       * inspelningar på bandet. Vi har alltså testat den sista luckan.
       * Filmen får inte plats på bandet.
       *)
      ok:=FALSE
    END
  END findFreeSpace;

BEGIN
  (* Proceduren leter upp den första luckan där filmen får plats. *)
  (* Börja med att testa om filmen får plats före första inspelningen. *)
  findFreeSpace(0,tape.recordings)
END RecordNewMovie;

(* + fler användbara procedurer... *)

END Tape.
```

5.

```
MODULE calc;
FROM InOut IMPORT ReadString,Done,Write,WriteString,WriteInt,WriteLn;
FROM Strings IMPORT Length,StrEq;
FROM Numbers IMPORT IsNumber,StringToNum;

PROCEDURE Fel(ord,fel:ARRAY OF CHAR);
BEGIN
  WriteString(ord); WriteString(fel); WriteLn
  HALT (* programmet stoppas *)
END Fel;

PROCEDURE Kalkylator;
  VAR it:INTEGER;

  PROCEDURE ReadOperand(VAR op:INTEGER);
    VAR ord:ARRAY [0..20] OF CHAR;
  BEGIN
    ReadString(ord);
    IF Done() THEN
      IF    IsNumber(ord)   THEN op:=StringToNum(ord)
      ELSIF StrEq(ord,"it") THEN op:=it
      ELSE Fel(ord," är ej en godkänd operand.")
      END
    END
  END ReadOperand;
```

```
  PROCEDURE ReadOperator(VAR op:CHAR);
    VAR ord: ARRAY [0..10] OF CHAR;
  BEGIN
    ReadString(ord);
    IF Done() THEN
      IF (Length(ord)#1) OR
         NOT ((op="+") OR (op="-") OR (op="*") OR (op="/") OR (op="="))
      THEN
        Fel(ord," är ej en godkänd operator.")
      END;
      op:=ord[0]
    END
  END ReadOperator;

  VAR operand1,operand2:INTEGER;
      operator:CHAR;

BEGIN (* Kalkylator *)
  it:=0;
  LOOP
    ReadOperand(operand1);
    IF NOT Done() THEN RETURN END;
    LOOP
      ReadOperator(operator);
      IF NOT Done() THEN RETURN END;
      IF operator="=" THEN EXIT END;
      ReadOperand(operand2);
      IF NOT Done() THEN RETURN END;
      CASE operator OF
        "+": operand1:=operand1+operand2
      | "-": operand1:=operand1-operand2
      | "*": operand1:=operand1*operand2
      | "/": operand1:=operand1 DIV operand2
      END
    END;
    it:=operand1;
    WriteInt(it,1); WriteLn
  END
END Kalkylator;

BEGIN
  WriteString("Välkommen till Thomas' Kalkylator"); WriteLn;
  Kalkylator
END calc.
```