INTERNATIONAL BACCALAUREATE
BACCALAURÉAT INTERNATIONAL
BACHILLERATO INTERNACIONAL

MARKSCHEME

November 2000

COMPUTER SCIENCE

Higher Level

Paper 2

16 pages

1. (a) When a "/" is found; (NO MARK for this)
the next character is tested ***[1 mark]*** or tests for two (consecutive) / s ***[1 mark]***
if it is a "/" the Boolean `COMMENT` is set to true ***[1 mark]***
if not `COMMENT` is set to false. ***[1 mark]***

(NOTE: for full marks the candidate must make clear that `COMMENT` is set to false, *i.e.* not just true is set.)

(b) The following statements are required (or similar):

```
if COMMENT then
   if POSITION = 2 then
      LINE <-- ""
   else
      LINE <-- copy(LINE,1,POSITION-2)
   endif
endif

if length(LINE) > 0 then
   output(NEWFILE) LINE
endif
```

(Award the marks as indicated below:)

– ***[4 marks]*** for reassigning `LINE` if a comment is found:
(– ***[2 marks]*** for the statement `LINE <-- copy(LINE,1,POSITION-2)` (***[1 mark]*** for a good, but incorrect, attempt at copying `LINE` using `POSITION`);
– ***[2 marks]*** for making `LINE` a nil string if a full line comment (***[1 mark]*** for any attempt at recognising this is not to be dealt with by copy, but incorrect). The candidate may set a flag and then the later **`if`** statement would include the **`not`** as well.)

– ***[3 marks]*** for transferring correct lines to `NEWFILE`:
(– ***[1 mark]*** for **`output`**`(NEWFILE)LINE` (or similar), it doesn't have to be precise `"PURE"`;
– ***[2 marks]*** for a correct if statement (***[1 mark]*** for any attempt at a test before writing to file, but incorrect).)

(c) The following is an algorithm to create the linked list:

```
procedure CREATELIST
   newtype NODE record
                   NUMBER integer
                   TEXT string
                   NEXT pointer->NODE
                endrecord

   declare STYLE is string file
   declare LINE string
   declare ROOT, TEMP pointer -> NODE

   open(STYLE)
   ROOT <- nil
   while not eof(STYLE) do
      input(STYLE) LINE
      allocate(TEMP)
      TEMP->NUMBER <- CONVERT(copy(LINE,1,4))
      TEMP->TEXT <- copy(LINE,5,length(LINE)-4)

      TEMP->NEXT <- ROOT
      ROOT <- TEMP
   endwhile
   close(STYLE)
endprocedure CREATELIST
```

(Award the marks as indicated below:)
(NOTE: exact `"PURE"` is not required; descriptive statements will gain full marks. Some examples are given within this markscheme. If in doubt, contact your team leader.)

– ***[2 marks]*** for initialising the head to nil:
  – (***[1 mark]*** using any pointer as a head, this can be given from a variable declaration, ***[1 mark]*** for setting it to nil, or similar, *e.g.* **`"empty"`** *etc.*)

– ***[1 mark]*** for correct loop, *i.e.* **`while not eof`**`(STYLE)` **`do`**
– ***[1 mark]*** for reading line correctly, *i.e* **`input`**`(STYLE)LINE` or similar (*e.g.* "read text/line/record from `STYLE` into `LINE`" *etc.*)
– ***[1 mark]*** for creating a dynamic variable, *i.e.* **`allocate`**`(TEMP)` or similar (*e.g.* "**`create`** `TEMP`", `"create a new record TEMP"` *etc.*)

– ***[5 marks]*** for correctly placing contents into `TEMP->NUMBER` and `TEMP->TEXT` (order is unimportant, *i.e.* `TEMP->TEXT` could be assigned first:
  – (***[1 mark]*** for using `TEMP->NUMBER` and `TEMP->TEXT` precisely, this is given in the question, so NO marks for `NUMBER` and `TEXT` only;
  – ***[2 marks]*** for using `CONVERT` correctly to extract the value from the comment with copy, (***[1 mark]*** for a good, but incorrect, attempt *e.g.* `"CONVERT`(first 4 characters)", `"CONVERT(LINE,1,4)"`, `CONVERT (LINE[1-4])"` *etc.*, NO marks for just `CONVERT` on its own, *e.g.* `TEMP->NUMBER <- CONVERT(LINE)`)
  – ***[2 marks]*** for assigning text part correctly, *i.e.* **`copy`**`(LINE,5,`**`length`**`(LINE)-4)`, (ACCEPT -5 at the end, even though incorrect, exam nerves and all, for ***[2 marks]***; also accept equivalent, precise statements not in `PURE`, *e.g.* "extract substring from character 5 to the end of the string") (***[1 mark]*** for a good, but incorrect, or vague attempt, *e.g.* "extract substring except first 5 characters").)
– ***[4 marks]*** for correctly assigning the pointers:
  – (***[2 marks]*** for making the record the head of the list, *i.e.* `TEMP->NEXT <- ROOT` (or equivalent); ***[1 mark]*** for any attempt at assigning `TEMP->NEXT`, but incorrectly;
  – ***[2 marks]*** for reassigning the head of the list, *i.e.* `ROOT <- TEMP`; (***[1 mark]*** for any attempt at assigning `ROOT`, but incorrectly).)

(d) The following is an algorithm to display the linked list:

```
procedure PRINTLIST(ref CURRENT pointer->NODE)
   if CURRENT # nil then
      PRINTLIST(CURRENT->NEXT)
   output CURRENT->TEXT
   endif
endprocedure PRINTLIST
```

(Award the marks as indicated below:)

– ***[2 marks]*** for a correct parameter:
  – (***[1 mark]*** for any parameter
  – ***[1 mark]*** for a good attempt at its type; accept just `"NODE"`)

– ***[3 marks]*** for correct terminator test:
  – (***[1 mark]*** for a correct if statement
  – ***[1 mark]*** for any attempt at a recursive call, *i.e.* the same procedure name is used, even if incorrectly
  – ***[1 mark]*** for a correct parameter, *i.e.* using `CURRENT->NEXT`; accept `CURRENT.NEXT`)

– ***[1 mark]*** for any attempt at an output statement for either field.

2. (a) (i) – Faster to read off card (than read from central database) ***[1 mark]***
– or could be used if link to central computer is down ***[1 mark]***

(ii) read from magnetic stripe ***[1 mark]***
decrypted (by software) ***[1 mark]***
compared with user input (at keypad) ***[1 mark]***

(b) *(Award marks as below, up to* ***[2 marks]*** *max:)*
easier to use a GUI because:
– just select from menus/options ***[1 mark]***
– without having to remember commands ***[1 mark]***
– quicker to select than type ***[1 mark]***

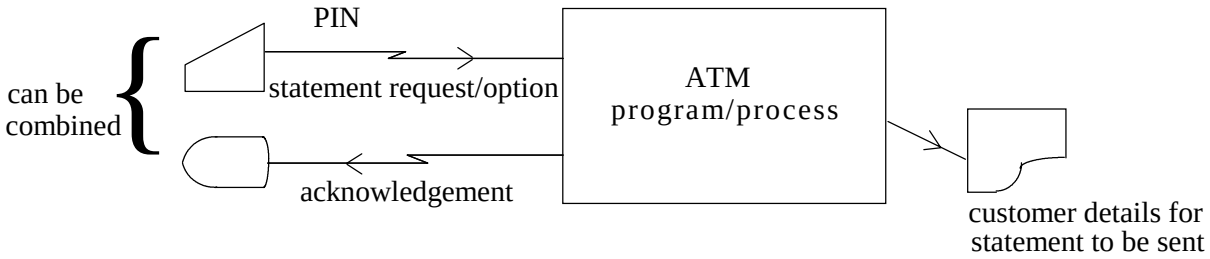
(c) *(Award* ***[1 mark]*** *for each point below up to* ***[2 marks]*** *max:)*
– may think his/her job is going to be lost
– may not trust process is fair/*i.e.* "it's going to happen anyway"/may not feel s/he has any influence
– if s/he reacts negatively, may fear getting the sack.

(d) more/larger/bigger storage space (do not accept "because it's bigger) ***[1 mark]***

(e) *(Award* ***[1 mark]*** *for a valid reason and* ***[1 mark]*** *for a suitable elaboration for two reasons up to a max of* ***[4 marks]****:)*
– fear of security, ***[1 mark]*** *e.g.* bank staff may accidentally see medical details ***[1 mark]***
– if reader of paramedics breaks down ***[1 mark]*** can't find information because not human-readable ***[1 mark]***
– can't see data ***[1 mark]*** so there might be a mistake ***[1 mark]***
– with all data on 1 card ***[1 mark]*** if lose it, lose loads of information ***[1 mark]***.

(f) (*Award* ***[1 mark]*** *for each point below up to* ***[5 marks]*** *max:)*
– both files are processed in one pass
– start reading (records) from both files
– compare key fields
– if the same, update master file record
– write updated record to new master file
– read both records
– if not same, write master file record to new master file
– read next master file record
– continue until all records in transaction file processed.

(g) The system flowchart is to ***[6 marks]*** **max**:



- ***[1 mark]*** for indicating telecomms link (either stated or ⟿)
- ***[3 marks]*** for user data (***[1 mark]*** for PIN, ***[1 mark]*** for request, ***[1 mark]*** for acknowledgement)
- ***[1 mark]*** for central program/process
- ***[2 marks]*** for output (***[1 mark]*** for an output box, ***[1 mark]*** for some indication of what it's for).

N.B: ***[7 marks]*** are available, give ***[6 marks]*** max.

(h) (i)
- central mainframe requests data from ATM ***[1 mark]***
- to see if data transfer is required ***[1 mark]***

(ii)
- ATM sends request to central mainframe ***[1 mark]***
- mainframe responds ***[1 mark]***

(iii)
- more secure ***[1 mark]***
- can't access from unknown lines ***[1 mark]***

3. (a) *(Award **[1 mark]** for each of the following points, up to a maximum of **[4 marks]**:)*

– Find/locate a free location/"blank" cell/empty string in the array;
– Give an error message if no free cell exists (and halt);
– Place "Lion" in the array (at this location);
– Make the link/ptr of "Lion" be the (current) PTR value of "Dog" OR the subscript/location of "Frog"
– Make the link of "Dog" be the subscript/location of "Lion"

(Accept for full marks the specific implementation of:)
– Place "Lion" `IN DATA[4]`
– Make `PTR[4] <-- 8` OR `PTR[4] <-- PTR[2]`
– Make `PTR[2] <-- 4`

(b) Possible algorithms are:

```
procedure REMOVE (ref ROOT integer; val ITEM string)
   declare POS integer

   if DATA[ROOT] = ITEM then
      ROOT <-- PTR[ROOT]
   else
      POS <-- ROOT
      while DATA[PTR[POS]] # ITEM do
         POS <-- PTR[POS]
      endwhile

      PTR[POS] <-- PTR[PTR[POS]]
   endif
endprocedure REMOVE
```

OR

```
procedure REMOVE (ref ROOT integer; val ITEM string)
   declare PREV,POS integer

   PREV <-- -1
   POS <-- ROOT

   while DATA[POS] # ITEM do
      PREV <-- POS
      POS <-- PTR[POS]
   endwhile

   if PREV = -1 then
      ROOT <-- PTR[ROOT]
   else
      PTR[PREV] <-- PTR[POS]
   endif
endprocedure REMOVE
```

– ***[1 mark]*** indicating that `ROOT` parameter or equivalent (probably `HEAD`) must be reference/variable;
– ***[2 marks]*** for dealing with the special case when `ITEM` is the first item to remove (***[1 mark]*** for any recognition that this needs to treated as a separate case, but not fully implemented);
– ***[2 marks]*** for correctly moving through the list until `ELEMENT` is found using `POS <-- PTR[POS]` (***[1 mark]*** for good, but incorrect attempt, that at least uses `PTR[POS]`;
– ***[2 marks]*** for correctly readjusting `PTR[POS]` (***[1 mark]*** for any attempt at this).

(c) *(Award **[1 mark]** for any of the following points, up to a maximum **[4 marks]**:)*

– (using arrays means that) memory must be set in advance;
– since memory is not allocated during run-time;
– which depends on the programmer knowing how large the list will be;
– (if list varies in length greatly) then a large amount of memory needs to be reserved which usually means it is underused;
– if list becomes too large for array, space cannot be increased, unlike dynamic structures;
– since dynamic structures allow space to be returned/disposed, memory management is more efficient.

(Allow brief points, e.g. "there is a fixed limit for the list", "memory not allocated in run-time" etc.)

4. (a) *(Award **[1 mark]** for each point as follows up to **[6 marks]** max:)*

The obvious implication is to use the question data as given, *i.e.*:
– Each record from the transaction file is read in turn
– The hash algorithm is applied to the key field
– An address is generated
– The block at that address is read
– The record is located by a (linear) search of records read
– (or for ***[1 mark]***, the record at the address generated is read)
– The record is updated with the new data from the transaction record
– The record is written back to the master file
– This continues until the end of the transaction file

However, the candidate may add new ideas, which are acceptable, for example:
– The master file may also have an index to access data
– The transaction file will be sorted using the same key
– Each record from the (sorted) transaction file is read in turn
– Each record from the master file index is read in turn
– If the keys match, the block at the address given is read
– The record is located by a (linear) search of the records read
– (or for ***[1 mark]***, the record at the address generated is read)
– The record is updated with the new data from the transaction record
– The record is written back to the master file
– This continues until the end of the transaction file

(Accept the above points in diagrammatic form.)

(b) (i) *(Award **[1 mark]** for any situation where the transaction file could be corrupted, e.g. fire, theft, flood, any physical/software created damage, even accept "if the transaction file gets corrupted".)*

*(Award **[2 marks]** for the actions that are required:)*
- ***[1 mark]*** for stating that a new transaction file will be created
- ***[1 mark]*** for identifying that it is copied from the backup

Only give ***[1 mark]*** for statement "the backup file is used"

(ii) *(Award marks as follows:)*
- ***[1 mark]*** for indication that a backup is made regularly
- ***[1 mark]*** for stating that it should be made after each change to the transaction file (or at least how the transaction file can be recreated).

Examples:
- 'Regular backups are made', 'Regular backups are made by copying transaction file', 'Backups are made every day' all get ***[1 mark]***.
- 'Backups are made after each change to the transaction file', 'Backups are made frequently during the day to minimise data loss', 'Backups are made daily, and the cheques kept to re-enter if the backup is needed' all get ***[2 marks]***.

(c) *(Award **[1 mark]** for each of the points below, depending on the method used:)*

- A list/index of all account numbers is used
- Each number is used/read in turn
- The hash algorithm is applied to the value
- The record read at the address generated and printed.

OR

- A separate (full) index with the address for all records is used
- Ordered/sorted on account number
- Each record is read/used to give the address of the main file record to read
- The record is read at the address from the index and printed.

5. (a) *(Award **[1 mark]** for any of the following points up to a maximum of **[4 marks]**:)*

– Cache memory is faster access memory than (RAM/ROM);
– frequently used instructions are stored in it (locality of reference);
– so the next instruction is retrieved faster;
– so the overall program runs faster.

(b) *(Award marks as follows:)*

– ***[1 mark]*** for a valid arithmetic instruction (*e.g.* add/subtract/multiply)
– ***[1 mark]*** for a valid logic instruction (*e.g.* and/or/not/shift)

Then a **maximum** of ***[4 marks]*** from the following:

– ***[1 mark]*** for the ALU carries out the operation based on a signal from the Control Unit/CU
– ***[1 mark]*** from decoding the (current) instruction/op-code
– ***[1 mark]*** the accumulator is a register
– ***[1 mark]*** where intermediate/final results
– ***[1 mark]*** of calculations/logic instructions are stored
– ***[1 mark]*** it has a direct link/bus to/from the ALU
– ***[1 mark]*** and is faster than using main memory/RAM.

(c) *(Award **[1 mark]** for a limitation, and **[1 mark]** for its implication:)*

– Since the program cannot be altered/changed ***[1 mark]***;
– any changes will require a new ROM chip to be inserted ***[1 mark]***;

– If the program gets corrupted (*e.g.* power surge) ***[1 mark]***;
– it cannot be debugged/"patched" ***[1 mark]***.

(d) *(Award **[1 mark]** for identifying that it is required to store any temporary data or data entered by the user or sensors, i.e. any of the following:)*

– To store temporary data;
– to store entered/user/sensor data;
– to store the current temperature;
– to store calculations made by the processor.

(Do NOT accept 'to store program instructions' (the question states all software is in ROM).)

(e) *(Award **[2 marks]** for a clear description, **[1 mark]** for a partial answer:)*

– a mantissa ***[1 mark]*** and an exponent ***[1 mark]*** can be used to store data;

– the register can have a fixed point ***[1 mark]*** set so that there are a set number of bits after the point ***[1 mark]***;

– a register can store the integer part ***[1 mark]*** and another can store the fraction part ***[1 mark]***;

(Do NOT award marks just for a statement similar to "use real/float values".)