Computer Science Department
1998 Examinations
D0a - MSc Basic Paper 1

Answer ALL questions

(The use of electronic calculators is not permitted in this examination)

1. Answer any **two** of parts a) b) c).

   a) i) The following C++ program will compile and when run will draw an isosceles triangle the size of which is given by the user. Comment on the "goodness" of this programming style and structure. Note that the line numbers are given for reference only.

```
 1: // isosceles.cc
 2: #include <iostream.h>
 3:
 4: int var1, var2, var3=1;  // declaring variables var1, var2, var3
 5: const char star = '\052';    // ASCII code for asterisk *
 6: float size;
 7:
 8: void main()
 9: {
10:   cout << "Enter size of triangle: ";
11:   cin >> size; cout << endl;var2=size+5;
12:   for (var1=0; var1<size; var1++) {
13:       var2 = var2 - 1;
14:       for (int i=0; i<var2; i++)  cout << ' ';
15:            for (int j=0; j<var3; ++j)  cout << star;
16:                 cout << endl;
17:            var3 = var3 + 2.0;
18:   }
19: }
```

[12 marks]

   ii) "*In C++, there is a strong relationship between pointers and arrays. Any operation that can be done by array subscripting can be achieved with pointers*".

   Explain this statement. Why is it generally preferable to use pointers?

[4 marks]

[Question 1 continued on next page]

[Question 1 continued]

b) i) The class `Calculator` represents a simple arithmetic calculator. It has a private data member `mem_store` of type `float`

It also has a number of public member functions that operate on the data:
`calculate, store, recall, clear`

The member function `calculate` has parameters `(float a, float b, char op)`. `op` may be `'*', '/', '+', '-'`. When called, the function should print to the screen the result of the appropriate expression. For example given a declaration of:

```
Calculator calc;
```

A call of `calc.calculate(10, 20, '+')` should print 30.

A call of `calc.store(78.4)` will store 78.4 in the data member `mem_store`

A call of `calc.recall()` will return the value stored in `mem_store`

A call of `calc.clear()` will "clear" `mem_store` to zero

Given the above information, define the class `Calculator` and its member functions. Provide a constructor function which should be used to initialise the data member. Note that your `calculate` function should have simple error checking, e.g. for division by zero. Minor errors in C++ will not be penalised.

[13 marks]

ii) Given the declarations

```
int arr[] = { 5, -4, 8, 23, 89, 1, 9, -987, 45 };
int *pi = &arr[4];
```

what is the value of the following C++ expressions?

```
A.   *arr + *pi
B.   *(pi+3)
C.   pi[-3]
D.   arr[--arr[6]]
```

[3 marks]

[Question 1 continued on next page]

[Question 1 continued]

c) The function `func` below can be used to produce a sequence of numbers

```
int func(int i)
{
    if (i <= 1) return 1;

    return (func(i-1) + func(i-2));
}
```

i) What would be the output of the following code fragment?

```
for (int i = 0; i < 5; i++) {
     cout << func(i) << "  ";
}
```

[6 marks]

ii) Given the output to i) above what is the relationship between the numbers in the sequence produced and what would be the next number in the sequence:

... 34 55

[2 marks]

iii) A recursive function will call itself either directly or indirectly. Given a call of `func(4)` how many additional calls of `func` will be made? Of this amount how many will be of the form `func(1)`?

[2 marks]

[Question 1c continued on next page]

[Question 1c continued]

iv) The code fragment below is incomplete and shows how `func` may be rewritten using iteration. The asterisks (*) represent incomplete code. Complete this version of `func` filling in the incomplete regions.

```
int func(int i)
{
     if (i <= 1) {
          return ****;
     }
     else {
          int prev1, prev2, current;
          prev1 = ****;
          prev2 = ****;
          for (int j = 2; j <= i; j++) {
               current = ****;
               prev2 = ****;
               prev1 = ****;
          }
     }
     return current;
}
```

[6 marks]

[CONTINUED]

2. Answer any **two** of parts a) b) c).

a) i) What is *virtual memory*? Describe how *paging* may be used to implement virtual memory. (Your answer should explain the rôle of *page tables* and *page faults*.)

[9 marks]

ii) Describe how an *Associative Memory Cache* may be used to improve the performance of a paged memory system.

[5 marks]

iii) Identify two circumstances in which it would be advantageous to allow two processes to share a page.

[2 marks]

b) i) The program below is written to run on a Motorola M68000-series processor which has a 32-bit address space. `a0` and `a1` are address registers, `d0` is a data register. The program purports to copy 100 16-bit words from an array starting at address `0x3000` to one starting at `0x4000`. There are several mistakes in the program. Identify these and explain how they should be corrected. [Note that the line numbers are for reference only]

```
1: loop: mov.w #0x3000, a0
2:       mov.w #0x4000, a1
3:       mov.w (a0), d0
4:       mov.w d0, (a1)
5:       add.l #1, a0
6:       add.l #1, a1
7:       cmp.l a0, #0x3064
8:       bne loop
9:       halt #0
```

[6 marks]

ii) Lines 3 and 4 in the program above are modified to use indexed addressing as follows:

```
3:       mov.w 0x3000(a0), d0
4:       mov.w d0, 0x4000(a0)
```

Provide the rest of the program to do the complete array copy. [You should use representative assembler mnemonics but you will not be penalised for minor syntax errors]

[10 marks]

[Question 2 continued on next page]

[Question 2 continued]

c) Typical implementations of C++ use a stack-based mechanism to implement procedure calls. The function below

```
int func(int a, int b){
     int i, j;
     ...
}
```

is called in the line:

```
x = func(5, 8);
```

where `x` is an integer. The diagram below shows a portion of a stack during the execution of this procedure call. (Note that the stack is 32 bits wide and grows down the page, all integers are 32-bit quantities)

Contents of stack

...	
12345	
8	
5	**<- Top of stack**
11111	
11111	
11111	
...	

i) Draw a diagram showing the state of the stack immediately after all the procedure initialisation is completed. Identify the top six items on the stack at that point and explain how they were placed on the stack.

[6 marks]

ii) Assuming register `a7` is used as a stack pointer illustrate how the stack "push" and "pop" instructions can be implemented using single assembly language instructions. State what addressing mode you are using.

[3 marks]

iii) It is common to use one register (`a5` say) as a *frame pointer* during procedure calls. Explain how a frame pointer may used to allow instructions in the compiled procedure to refer to the parameters and local variables.

[7 marks]

3. Answer any **two** of parts a) b) c).

a) i) What is the difference between *interrupt driven input/output* and *polled input/output*? Under what circumstances might each be appropriate?

[6 marks]

ii) Explain the roles of *daisy chaining*, *interrupt vectors and return from interrupt instructions* in interrupt processing.

[10 marks]

b) A file called *fred* is referenced from a directory called *jim.* The file occupies four disc blocks, each of 1024 bytes and numbered 1230, 4961, 2314 and 1102 in that order.

i) Explain how the directory and file would be organised in an MS-DOS-style file store.

[4 marks]

ii) Explain how the directory and file would be organised in a Unix-style file store.

[4 marks]

iii) Both MS-DOS and Unix support *random access* to disc files. Explain what this means and outline the steps that would be taken in the two systems when accessing byte number 2050 of the file *fred.* Note any rôle cacheing plays in these procedures.

[4 marks]

iv) MS-DOS stores file attributes in the directory, Unix stores them in the I-node. What is the impact of these design decisions when a file has multiple names (i.e. multiple directory entries).

[4 marks]

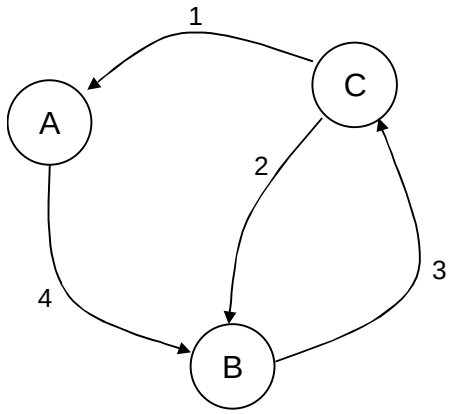
[Question 3 continued on next page]

[Question 3 continued]

c) i) In the context of operating systems, what is *pre-emptive scheduling*? Give examples of situations in which pre-emptive scheduling is A. desirable, B. undesirable

[5 marks]

ii) The diagram below shows state changes which can occur during the life-cycle of a process.



What are the states A, B and C? Give examples of circumstances in which each of the transitions 1, 2, 3 and 4 take place.

[9 marks]

iii) Is it possible for transition 2 to take place in a non-pre-emptive system?

[2 marks]

[END OF PAPER]