Ollscoil na hÉireann, Gaillimh
National University of Ireland, Galway

GX 2008

Semester II Examinations, 2003/2004

Exam Code(s)	1BS1, 1CS1, 1EL1
Exam(s)	**First Science**
Module Code(s)	CS102
Module(s)	Computer Science
Paper No.	SUMMER
Repeat Paper	
Special Paper	
External Examiner(s)	Prof. P. Nixon;
Internal Examiner(s)	Prof. G. Lyons,
	Dr. G. Pfeiffer,
	Dr. R. Butler,
	Mr. A. Reilly,
	Dr. B. Gleeson.
Instructions:	Attempt *SIX* questions, at least *ONE* from each section. Use a *SEPARATE ANSWER BOOK* for each section.
Duration	*THREE* hours
No. of Answer books	*THREE*
Requirements:	
Handout	_____
MCQ	_____
Statistical Tables	1
Graph Paper	_____
Log Graph Paper	_____
Other Material	_____
No. of Pages	5 PAGES (Excluding Front Page)
Department(s)	MATHEMATICS,
	EXPERIMENTAL PHYSICS,
	INFORMATION TECHNOLOGY
	MATHEMATICAL PHYSICS

Section A

1. (a) Convert the decimal values 37_{10} and 183_{10} into their binary number form.

   (b) Determine the 8-bit 2's-complement form of the decimal numbers -45_{10} and -110_{10}.

   (c) Using 2's-complement representation, evaluate the following:

   i) $37_{10} + (-45_{10}) = ?$,
   ii) $183_{10} + (-110_{10}) = ?$.

   (d) Explain how the Hamming Code works. Determine the Hamming Code for the 7-bit binary equivalent of 61_{10}.

2. (a) State *BOTH* of deMorgan's Laws for Boolean operators, and prove *EITHER* law using a truth table.

   (b) Simplify the following Boolean expressions:

   i) (X NAND X) NAND (Y NAND Y);
   ii) (X NOR Y) NOR (X NOR Y).

   (c) Discuss the logical design and operation of *ANY TWO* of the following electronic circuits:

   i) binary decoder,
   ii) multiplexor,
   iii) data latch.

3. (a) Draw a diagram of the outline logical design of a simple Central Processing Unit (CPU), labelling all of its different components (buses, registers etc.).

   (b) Illustrate the procedure by which a Fetch/Execute cycle might be implemented by the CPU.

   (c) Explain the meaning and operation of *ANY EIGHT* of the following ten machine code operations used in the CS1 machine/simulator:

   1. JZA,
   2. JPA,
   3. JNA,
   4. LDX,
   5. STX,
   6. ADX,
   7. SBX,
   8. SJX,
   9. JZX,
   10. DJX.

Section B

4. (a) Explain the following terms:
   i) *token*,
   ii) *identifier*,
   iii) *comment*,
   iv) *expression*, and
   v) *statement*.

   (b) Illustrate by example how the following operators work:
   i) &&,
   ii) &,
   iii) /=,
   iv) ++, and
   v) %.

   (c) Assuming the declarations and initializations:

   int a = 3, b = 1, c = 4;

   give the values of the following expressions, if they are legal:
   i) -b < (a < c),
   ii) 2 - c % a + b, and
   iii) ++a * c- -.

5. (a) Rewrite the following program using proper indentation, comments and descriptive identifiers to make it more readable and well documented:

```
int main(void ){float qx,
zz,
tt;printf("gimme 3" );scanf
( "%f%f %f",&qx,&zz
,&tt);printf("averageis=%f",
(qx+tt+zz)/3.0);return
0;}
```

   (b) Describe the process of function invocation in a C program.

6. Write a short program that uses a *recursive function* to print out the integer n, digit by digit, if n is given as a command line argument, as follows. As first things in the file, include the header files stdio.h and stdlib.h. Then define a function printd() which takes an integer argument n, returns no value and whose body consists of the following three statements:

   (a) If the argument n is negative, use putchar() to print out a minus sign and replace n by –n;

   (b) Call printd() recursively with argument n/10 provided this is not zero;

   (c) Use putchar() to print the last digit of n as n % 10 + '0'.

   Then define a main() function that provides access to the command line through argc and argv such that

   (a) an error message is printed unless the command line consists of exactly two words, the name of the program and a command line argument;

   (b) printd() is applied to the command line argument after it has been converted to an integer by the function atoi();

   (c) a newline character is printed out.

Section C

7. Describe briefly, using diagrams and/or examples where appropriate, ANY THREE of the following:

   (a) Multimedia on the World Wide Web (WWW);

   (b) Database Management System (DBMS)

   (c) Magnetic and Optical Storage Devices;

   (d) Internet Connection Methods;

   (e) Operating Systems;

   (f) Data Mining.

8. (a) Explain why pointers are useful, and give one practical example.

(b) Assume that we have a C-programme with the following code

```
int a, b, c, *x, *y, *z;
z = &a;
a = 10;
c = a - 7;
x = &b;
*x = a/4 + c;
y = z;
*y += 1;
b -= c;
```

After the code has executed, what values will a, b, c, *x, *y and *z have. (Try to draw a flow diagram of this sequence of instructions).

(c) The following programme takes NUM integers from the user, and puts them in ascending order of size, using a standard bubble sort:

```
// Bubble Sorter
# include < stdio.h >
# define NUM 10
main(void)
{
    int i, j, array[NUM];
    void swap(int *, int *);
    printf("Input %d elements.\n", NUM);
    for (i = 0; i < NUM; i++)
        scanf("%d", array[i]);
    for (i = 0; i < NUM-1; i++)
        for (j = 0; j < NUM-1; j++)
            if (array[i] > array[i+1])
                swap(&array[i], &array[i+1]);
    printf("The array elements in order are:\n\n");
    for (i = 0; i < NUM; i++)
        printf("Element %d is %d.\n", i+1, array[i]);
}
```

Write, and comment, the function **void** swap(**int** *x, **int** *y).

(d) Suppose now, that the programme is rewritten in the following form:

```
// Bubble Sorter
# include < stdio.h >
# define NUM 10
void main(void)
{
    int i, j, array[NUM];
    void swap(int *, int *);
    void bubblesort(int *, int);
    printf("Input %d elements.\n", NUM);
    for (i = 0; i < NUM; i++)
        scanf("%d", array[i]);
    bubblesort(array, NUM);
    printf("The array elements in order are:\n\n");
    for (i = 0; i < NUM; i++)
        printf("Element %d is %d.\n", i+1, array[i]);
}
```

Write, and comment, the function **void** bubblesort(**int** *x, **int**).

9. (a) Explain why structures are useful, and give one practical example.

(b) Write a commented programme, using the following structure type:

```
typedef struct
{
    float re;
    float im;
} complex;
```

to input two complex numbers, and output their sum and product:

i) where the entire programme is contained in main(**void**);

ii) where addition and multiplication of the complex numbers is executed by **void** add(**complex** *x, **complex** *y, **complex** *z) and **void** multiply(**complex** *x, **complex** *y, **complex** *z), respectively.