King's College London

UNIVERSITY OF LONDON

This paper is part of an examination of the College counting towards the award of a degree. Examinations are governed by the College Regulations under the authority of the Academic Board.

B.Sc. EXAMINATION

CP/1710 Computing for Physical Sciences

Summer 2003

Time allowed: THREE Hours

Candidates must answer SIX parts of SECTION A, and TWO questions from SECTION B.

The approximate mark for each part of a question is indicated in square brackets.

You must not use your own calculator for this paper.
Where necessary, a College calculator will have been supplied.

TURN OVER WHEN INSTRUCTED

pi $\pi = 3.1415927$

Ensure that any *C* code you write in answer to the questions below is clearly laid out, and contains sufficient comments for the reader to understand the code.

SECTION A – Answer SIX parts of this section

1.1) Identify which of the following 10 names are *not* valid for *C* variables, giving a reason in each case.

```
CASE          null
date-stamp    time&date
default       UK£
Double        VOID
not4now       2nd_choice
```

[7 marks]

1.2) *C* variables of type `float, long` and `unsigned long` all use 4 bytes of memory to store numerical values. Briefly describe the differences between these data types, and the way in which numerical values are stored in the 4 bytes used.

[7 marks]

1.3) Inside a *C* function some variables are declared with the following statements:

```
const char a='h';
unsigned char b=200;
char e[]="hello";
char *f=e;
```

Distinguish clearly the consequences of these different declarations for each of the variables `a, b, e, f`.

[7 marks]

1.4) Explain briefly why it can be helpful to use pointer variables as the arguments to *C* functions.

[7 marks]

1.5) What information do the following *C* statements provide to the compiler?

```
float xy_position[100][2]= {0};
int Midpt_3d( long num_pts, double x[], double y[], double z[],
              double *cx, double *cy, double *cz);
```

[7 marks]

1.6) A structure of type `box` is defined by the *C* code fragment below.

```
struct box
{  float width;   // in metres
   float height;  // in metres
   float depth;   // in metres
};
```

A *C* function `Box_volume()` is intended to accept a structure of type `box` as an argument, and to return the value of the box's volume to the calling program. It should return the value zero if any of the box dimensions are negative.

Provide a *definition* for the function `Box_volume()` that will meet the requirements given above.

[7 marks]

1.7) Briefly summarise the actions carried out by the following *C* program, and describe the output that is produced.

```
#include <stdio.h>  // needed for I/O
int main()
{  int x[5]= {1, 2, 1, 2, 2};
   int y[5]= {0};
   int i;
   for (i=0 ; i<5 ; i++)
   {  if (i > 0)
      {  y[i]= (x[i] > x[i-1])? 1: -1 ;
         if (x[i] == x[i-1]) y[i]= 0;
      }
      printf("x[%d]= %2d, y[%d]= %2d \n", i, x[i], i, y[i]);
   }
   return 0;
}
```

[7 marks]

1.8) A *C* program initialises the variables `a, b, c, d` and `x` in the following way.

```
int a=5, *b;
b=&a;
float c=7.0, x=4.0;
float *d;
d=&c;
```

The following three *C* statements appear consecutively within the program.

```
x = a + (*b);
x = (++a) * (*d);
x *= (*d);
```

What are the values of `x` after each of the three *C* statements has been executed?

[7 marks]

SECTION B – Answer TWO questions

2(a) A triangle has sides of lengths a, b and c. The cosine rule allows the angle A, opposite to the side of length a, to be calculated using the formula

$$\cos A = \frac{b^2 + c^2 - a^2}{2bc}$$

Define a C function that will accept values for the lengths a, b and c as parameters, and will return the angle A (in radians) to the calling program. Make sure that your function will issue a warning message, and return the value 0, when an invalid set of parameters is supplied.

Note: Assume that the C function `acos()` is declared in the library `<math.h>` as follows:

```
double acos( double x);
/* Returns the inverse cosine (arc-cosine) of x (in radians).
 * An error occurs when x does not lie
 * in the interval from -1 to 1.
 */
```

[15 marks]

(b) Write a C program that will carry out the tasks listed below.

(i) Repeatedly prompt the user to enter a set of three positive values for the sides of a triangle. The user should be told to enter the three values on a single line, separated by commas. The program should end if any of the three values entered is negative.

(ii) For each set of three values entered that correspond to a valid triangle, the program should calculate the size of the three opposing angles of the triangle (in degrees), by making suitable calls to the function you defined in part (a).

(iii) For each triangle, the program should print out to the computer screen a two-line message in the following style:

```
For a triangle with sides 1.00, 1.00, 1.41 metres,
the opposing angles are   45.0, 45.0, 90.0 degrees.
```

[15 marks]

3) The inverse tangent function can be represented by an infinite series of the form

$$\tan^{-1}(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots + (-1)^n \frac{x^{2n+1}}{2n+1} + \ldots$$

(a) Define a function in C that will return the value of $\tan^{-1}(x)$ using the series expression given above. The function you define should have the following declaration:

```
double Inverse_tan( double x, int n);
```

where the parameter `n` specifies the number of terms to be used in the series approximation.

[12 marks]

(b) Write a `main()` program in C that will carry out the following tasks:

(i) Repeatedly prompt the user to supply a value for the variable `x`, exiting when the value zero is supplied.

(ii) Print out the value of $\tan^{-1}(x)$, calculated by the function `Inverse_tan()`, for all values of `n`≤ 20,

(iii) Calculate $\tan^{-1}(x)$ using the C library function `atan(x)`, and print this value on the line following the results of the series approximation.

Note: Assume that the C library function `atan()` is declared in the library `<math.h>` as follows:

```
double atan( double x);
/* Returns the value (in radians) for the
   inverse tangent (arc-tangent) of x */
```

[12 marks]

(c) Comment on the accuracy of the approximation given by the function `Inverse_tan(x, 20)` for $|x| \leq 1$.

[6 marks]

4(a) With the aid of a sketch, give a graphical interpretation of the **Newton-Raphson** method for locating a root of the equation $f(x) = 0$, where $f(x)$ is a continuous real function. Explain briefly why the Newton-Raphson method may sometimes fail to locate the desired root.

[7 marks]

(b) Identify the initial conditions that must apply for the **bisection** method to be *guaranteed* to find a root of the same function $f(x)$. Explain briefly why the bisection method is said to converge *linearly*.

[5 marks]

(c) Determine a suitable interval within which you expect to find the single positive real root of the equation

$$3x^3 + 5x^2 - 16x - 28 = 0$$

[2 marks]

(d) Write a *C* program that uses the bisection method to find this root of the equation to an accuracy of 4 decimal places, and then prints out the value found.

[12 marks]

(e) The equation has another root at $x = -2$. Explain why it would not be possible to use the bisection method to find this root. Comment briefly on whether it would be possible to use the Newton-Raphson method instead.

[4 marks]

5) It is often useful to apply a linear filter to a set of experimental data.

(a) Define a function in C, with the declaration

```
void Filter_3pt( float *input, float *output, int num_pts,
                 float a1, float a2, float a3);
```

that will apply a three-point linear filter to the `input` data. The first element and the last element of the `output` data should be the same as the first and last elements of the `input` data, respectively, but every other element of `output` should be calculated according to the C expression

```
output[n] = (a1*input[n-1]) + (a2*input[n]) + (a3*input[n+1]);
```

[6 marks]

(b) A set of experimental data is stored as formatted values in a plain text file called `signal.txt`. The data consist of 120 real values, stored as a single column with 120 lines of text. Write a C program that will carry out the following tasks:

(i) Read the data from the specified file into a one-dimensional array named `signal_in`.

(ii) Apply the function `Filter_3pt()` to the `signal_in` data using coefficients

$$\texttt{a1} = \texttt{a2} = \texttt{a3} = \frac{1}{3}$$

placing the results in an array named `signal_out`.

(iii) Write out both the `signal_in` and the `signal_out` data as plain text to a file called `filtered_signal.txt`. The output should be formatted so that the input and filtered data appear as two neatly aligned columns.

[12 marks]

(c) Briefly describe the effects of using the filter described above on the data read in from the file `signal.txt`. Suggest why this particular filter may be of use.

[6 marks]

(d) Consider an array of signal data declared and initialised by the C statement

```
float s[10]={ 0, 0, 0, 1, 1, 1, 1, 0, 0, 0};
```

and a linear filter with coefficients

$$\texttt{a1} = -\frac{1}{2},\ \texttt{a2} = 0,\ \texttt{a3} = \frac{1}{2}$$

Describe the effect of applying this filter to the data in array `s[]`, and suggest a general use for this type of filter.

[6 marks]