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

CP1710 Computing for Physical Sciences

Examiner: Dr. G.R. Morrison

Summer 2007

Time allowed: THREE Hours

Candidates may answer as many parts as they wish from SECTION A, but the total mark for this section will be capped at 40.

Candidates should answer no more than TWO questions from SECTION B. No credit will be given for answering a further question from this section.

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

You may use a College-approved calculator for this paper.

TURN OVER WHEN INSTRUCTED

Physical Constants

Permittivity of free space	ϵ_0	=	8.854×10^{-12}	$\mathrm{F\,m^{-1}}$
Permeability of free space	μ_0	=	$4\pi \times 10^{-7}$	$\mathrm{H\,m^{-1}}$
Speed of light in free space	c	=	2.998×10^{8}	$\mathrm{m\,s^{-1}}$
Gravitational constant	G	=	6.673×10^{-11}	$\mathrm{N\,m^2\,kg^{-2}}$
Elementary charge	e	=	1.602×10^{-19}	C
Electron rest mass	m_e	=	9.109×10^{-31}	kg
Unified atomic mass unit	m_u	=	1.661×10^{-27}	kg
Proton rest mass	m_p	=	1.673×10^{-27}	kg
Neutron rest mass	m_n	=	1.675×10^{-27}	kg
Planck constant	h	=	6.626×10^{-34}	J s
Boltzmann constant	k_B	=	1.381×10^{-23}	$\mathrm{J\,K^{-1}}$
Stefan-Boltzmann constant	σ	=	5.670×10^{-8}	$\mathrm{W\,m^{-2}\,K^{-4}}$
Gas constant	R	=	8.314	$\mathrm{J\,mol^{-1}\,K^{-1}}$
Avogadro constant	N_A	=	6.022×10^{23}	$\mathrm{mol^{-1}}$
Molar volume of ideal gas at STP		=	2.241×10^{-2}	$\mathrm{m^3}$
One standard atmosphere	P_0	=	1.013×10^{5}	$\mathrm{N\,m^{-2}}$

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

SECTION A

Answer as many parts of this section as you wish.

The final mark for this section will be capped at 40.

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

```
First_value     Switch
3rd_value       break
char            fifty%
US_$            a999999
FLOAT           infinity
```

[5 marks]

1.2) The following three statements appear *before* the `main()` function of a *C* program is defined:

```
#include <math.h>
#define FOURPI 12.5663706
float Average( int a, int b, int c);
```

Briefly describe the consequences of each statement.

[7 marks]

1.3) Rewrite the following *C* program so that it makes use of a `while` loop instead of a `for` loop, but will produce identical output.

```
#include <stdio.h>    // Needed for I/O functions
//
int main()
{
   int i;
   for ( i = -10; i <= 0; i++)
      printf( "Time until launch: % 3d s \n", i);
   return 0;
}
```

[5 marks]

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

```
struct polar { float r; float theta; float phi;};
struct polar location;
struct polar Convert_coord( float x, float y, float z);
```

[7 marks]

1.5) A temperature f in degrees Fahrenheit can be converted to a temperature t in kelvin using the relation

$$9(t - 273.15) = 5(f - 32).$$

Provide a declaration and a definition for a *C* function that will accept a temperature in degrees Fahrenheit as an argument, and return the equivalent temperature in kelvin to the calling routine.

[7 marks]

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

```
struct sphere
{  float density;     // in kg per cubic metre
   float radius;      // in metres
};
```

Define a function in *C* that will expect a single argument that is a structure of type `sphere`, and which will return the mass of the sphere to the calling program.

[6 marks]

1.7) The *C* variables `a, b, c, x, y, z` are declared and initialised as follows:

```
int a=0, b=1, c=3;
long x=10, y=4, z[2]={6, 8};
```

Determine the values of `x` after each of the following *C* statements has been executed.

```
(i)      x += (y * c);
(ii)     x %= y;
(iii)    x *= (y + (b<c));
(iv)     x -= ( z[a]<z[b]? z[a]: z[b]);
```

[8 marks]

Note: Treat each part of this question independently, the statements are not consecutive steps in the same program.

1.8) 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]

1.9) Describe the **straight insertion** method of sorting an array of N values into order. Straight insertion sorting is referred to as an N^2 method: briefly explain what this means, and state one consequence of this property.

[8 marks]

SECTION B – Answer TWO questions

2a) The nth-order Bessel function of the first kind $J_n(x)$ is given by the infinite series

$$J_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \left(-\frac{1}{4}\right)^k \frac{x^{2k}}{k!\,(k+n)!}$$

where n is an integer, and x is a real number.

Define a C function with the declaration

```
float BesselJ( int n, float x, unsigned m);
```

that can be used to calculate $J_n(x)$ using the first m terms of the infinite series. Ensure that the function you define will return 0 when $m = 0$.

[15 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 variables `n, x` and `m`, exiting when the value zero is supplied for `m`.

(ii) Print out the value of $J_n(x)$, calculated by the function `BesselJ(n, x, i)` that you have defined, for all integer values `i` in the range $0 \leq$ `i` $\leq$ `m`,

(iii) Calculate $J_n(x)$ using the C library function `jn(n, x)`, and print this value on the line following the results of the series approximation.

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

```
double jn( int n, double x);  /* Returns the value of the
                                 n-th order Bessel function
                                 of the first kind, Jn(x) */
```

[15 marks]

3a) Define a function in C, with the declaration

```
void Filter5pt( float input[], float output[],
                int numpts, float a[]);
```

that will apply a five-point linear filter to the values held in the array `input`, storing the results in the array `output`. The array `a` is a five-element array of filter coefficients. The first two elements and the last two elements of the `output` data should be the same as the first two and last two elements of the `input` data, respectively, but every other element of `output` should be calculated according to the C expression

```
output[n] =  (a[0]*input[n-2]) + (a[1]*input[n-1])
           + (a[2]*input[n]) + (a[3]*input[n+1])
           + (a[4]*input[n+2]);
```

[7 marks]

b) A set of experimental data is stored as formatted values in a plain text file called `data50.txt`. The data consist of 50 real values, stored as a single row with a comma immediately following every value.

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 a 5-point moving-average filter to the data to the `signal_in` data, placing the results in an array named `signal_smooth`.

(iii) Filter coefficients with values

$$\mathtt{a}[0] = \mathtt{a}[2] = \mathtt{a}[4] = 0, \quad \mathtt{a}[1] = -\frac{1}{2}, \quad \mathtt{a}[3] = \frac{1}{2}$$

result in an output related to the gradient of the input. Apply this gradient filter to the values in the array `signal_smooth`, storing the results in an array named `signal_slope`.

(iv) Append the values of the `signal_smooth` data and the `signal_slope` to the same plain text file `data50.txt` from which the original `signal_in` data were read. The output should be formatted so that the `signal_smooth` and the `signal_slope` values are each written as 50 comma-separated values on single lines, immediately following the row of `signal_in` values.

[15 marks]

c) Determine the coefficients for a single linear filter that could be used to produce the same effect as the successive application of the moving-average and gradient filters used above.

Comment briefly on whether the order in which the smoothing and gradient filters are applied to the data affects the outcome.

[8 marks]

4) A real 2×2 matrix $\mathbf{M}$, its determinant, and its inverse can be written as

$$\mathbf{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad \det(\mathbf{M}) = ad - bc, \quad \mathbf{M}^{-1} = \frac{1}{\det(\mathbf{M})} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

In a *C* program the matrix $\mathbf{M}$ can be represented as a two-dimensional array. Write a `main()` *C* program that will accomplish the following tasks:

(i) Create an array to store the real 2×2 matrix $\mathbf{M}$, and prompt the user to supply values for all the entries in the array.

(ii) Determine whether or not the inverse of matrix $\mathbf{M}$ exists, and print a suitable message to the screen.

(iii) If the inverse matrix exists, then calculate it, and print its values to the screen, ensuring that the output appears as two rows with two neatly aligned columns. (Do *not* attempt to print the parentheses that enclose the matrix.)

(iv) Calculate the *transpose* of the matrix $\mathbf{M}$, and store it as a second matrix $\mathbf{T}$.

(v) Calculate the matrix $\mathbf{A}$ given by the expression $\mathbf{A} = \mathbf{M} \cdot \mathbf{T} - \mathbf{T} \cdot \mathbf{M}$, where the central dot $(\cdot)$ denotes matrix multiplication, and print the matrix $\mathbf{A}$ to the screen, again with two neatly aligned rows and columns.

[30 marks]