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.

M.Sci. EXAMINATION

CP/4731 C and C++ programming for physicists

Summer 2000

Time allowed: THREE Hours

Candidates must answer any THREE questions. No credit will be given for attempting a further question.

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

Good answers to questions will include plans and explanations in addition to sections of C or C++ code.

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

TURN OVER WHEN INSTRUCTED

Answer THREE questions

1) Write a short function in C, called `sinc(x)`, with a single positive argument *x*, which returns the value of $\frac{\sin x}{x}$ to 7 decimal places, by using the series:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \cdots$$

You should use enough terms of the series to ensure that the series converges to the accuracy required. Although this series is valid for all values of *x*, you should exploit the fact that sin *x* is periodic to ensure that the series converges quickly even when |*x*| is very large.

[20 marks]

2) The fourth-order Runge-Kutta method of solving a first order differential equation $\frac{dv}{dt} = f(v,t)$, numerically, is given by the following set of equations:

$$k_1 = \Delta t. f(t_n, v_n)$$
$$k_2 = \Delta t. f(t_n + 0.5\Delta t, v_n + 0.5k_1)$$
$$k_3 = \Delta t. f(t_n + 0.5\Delta t, v_n + 0.5k_2)$$
$$k_4 = \Delta t. f(t_n + \Delta t, v_n + k_3)$$
$$k = (k_1 + 2k_2 + 2k_3 + k_4)/6$$
$$x_{n+1} = t_n + \Delta t, \quad v_{n+1} = v_n + k$$

That is, if $v_n(t_n)$ is known, then these equations can be used to step forward by Δ*t* and calculate $v_{n+1}(t_{n+1})$, and hence find *v*(*t*) for all *t*.

Write a short program in C or C++ which calculates the velocity of an object dropped into a viscous medium, *v*(*t*), such that $\frac{dv}{dt} = av^n - g$ starting from *v* = 0 at *t* = 0 until *t* = 20 s, for given constants *a*, *n* and *g* = 9.81 ms^{-2}. It should print out the velocity every 0.1 s.

[20 marks]

3) Design a class which deals with *n*×*m* matrices. It should include a default and a real constructor, which initialise the elements of the matrix to zero, a destructor and functions to set and access elements of the matrix. It should allocate space for the matrix dynamically.

Show how the + operator could be overloaded within your class of matrices to specify matrix addition. It should print an error if the matrices are of dimensions which make addition impossible.

[20 marks]

4) Design a hierarchy of classes in C++ with the following properties:
`point` contains the 3-D coordinates of a point (*x*, *y*, *z*),
`shape` contains a position (a point).
Inheriting from **`shape`**:
`sphere` contains the position of its centre and a radius,
`cube` contains the position of its centre and the length of one side,
`cuboid` contains the properties of **`cube`**, but also has the lengths of the two additional sides.
Each class has overloaded constructors, a destructor, and a (virtual) function which returns its volume.

[20 marks]

SEE NEXT PAGE

5) The following C++ code sets up a linked list. Explain what it is, how it works, and what each of the functions does.

[10 marks]

```
struct element { float number; element *next; } ;

class linked_list
{
public:
      linked_list(){ first = last = new element;};
      ~linked_list();
      void Add(float);
      void MoveStart(){ ptr = first;}
      int MoveOK(){ return ptr != last;}
      float Move();
private:
      element *first, *last, *ptr;
};

linked_list :: ~linked_list()
{
      element *p;
      while (first != last)
      {
            p = first;
            first = first->next;
            delete p;
      }
      delete first;
}

void linked_list ::Add(float another)
{
      last->number = another;
      last = last->next = new element;
}

float linked_list ::Move()
{
      float i = ptr->number;
      ptr=ptr->next;
      return i;
}
```

Add a function which would calculate the mean of the numbers stored in the linked list. Write the main function to read 10 floats into the list and calculate their mean.

[10 marks]

FINAL PAGE