1993 Paper 5 Question 5

Programming in C

You have a C compiler which is ANSI conforming in all respects except that it has no facility for the definition, declaration or use of standard C structures. Outline a set of routines written in this language to provide a mechanism for handling structures.

Your solution should contain the following:

(a) function prototypes for each of the routines [10 marks]

(b) a few sentences describing the behaviour of each function [10 marks]

Note: no code other than the prototypes is required.

1993 Paper 6 Question 5

Programming in C

Two identical packs of ordinary playing cards (52 different cards in a pack) are shuffled and placed face downwards on a table. Two players then play a game of *Snap*. Each player is allocated one pack and at each turn in the game one card in each pack is turned up and the two upturned cards are compared. If the cards are the same (i.e. match in every respect) a *snap-turn* is declared. A game ends when all 52 pairs have been compared.

Write a C program which will simulate the game for the purposes of determining the probability of there being at least one snap-turn in a game. [20 marks]

Note: you may assume the existence of a random number generator but must state its properties.

1994 Paper 5 Question 5

Programming in C

Write a program in C which, given two integer inputs J and K, will output the combinations of J things partitioned into K groups. For example, if $J = 5$ and $K = 3$, the output would be:

```
(5,0,0)
(4,1,0)
(3,2,0)
(3,1,1)
(2,2,1)
```

[20 marks]

1994 Paper 6 Question 5

Programming in C

Write a program in C which can solve cryptarithmetic puzzles in the format of the sum of two words. For example, given the input

```
 SEND
+MORE
-----
MONEY
```

the program would output

```
 9567
+1085
-----
10652
```

N.B. Each letter represents a different digit. [20 marks]

1995 Paper 5 Question 5

Programming in C and C++

Students have been set a programming exercise where they are expected to write a subroutine to perform a certain operation, and they have been told that they should write it in the language C. One of the offerings submitted for assessment is the following. The student involved, who had read a book by Dijkstra but not understood it, proudly proclaims that of course this program has not actually been tried on a computer.

Explain what the program is (probably) supposed to do, and identify as many problems with it as you can, given that in this examination questions are expected to be completed in around 30 minutes.

```
typedef unsigned long int thing;

#define swap(p,q) v = p; p = q; q = v;

void fast(thing a[], int left, int write)
{                     /**********************
    int i, j;          * Declare variobles! *
    thing v;           **********************/
    if (write-left > 1)
    {   v = a[write];
        i = left, j = write;
        for (;;)
        {   while (a[++i] < v};
            while (a[--j] >= v);
            if (i < j) swap(a[i], a[j]);
            else break;
        }
        fast(a, left, i-1);  // re-curse here.
        fast(a, i, write)
    }
}
```

[20 marks]

1995 Paper 6 Question 5

Programming in C and C++

Write brief notes on *four* of the following aspects of the language C++. In some of the cases it may be appropriate to compare what C++ does with the situation in other programming languages such as C, ML or Modula-3.

(a) Overloaded functions and operators.

(b) Templates as a way of achieving operations on lists where the types of the items in the lists must be kept flexible.

(c) Consistency checks when a program is kept in several files and these are compiled at different times.

(d) Inheritance and virtual functions.

(e) The problems of writing code that is portable from one host machine to another.

[5 marks each]

1996 Paper 5 Question 5

Programming in C and C++

For *five* of the following C or C++ features write a very short fragment of code (perhaps 2 or 3 lines will suffice in most cases) that illustrates the syntax involved. In each case explain very briefly what your example achieves.

(a) preprocessor macros and conditional compilation

(b) casts that convert from one pointer type to another

(c) C and C++ style comments

(d) the declaration of a simple C++ class

(e) overloading the operator '`+`'

(f) the C `setjmp` function

(g) the `switch` statement, including a `default` label

[4 marks each]

1996 Paper 6 Question 5

Programming in C and C++

A grand debate is being planned by a society that has among its members a large number of computer professionals and working programmers. The motion to be put is "That the languages C and C++ should be consigned to outer darkness and their use banned in all serious computer projects". Prepare as your answer to this question a briefing document that could explain to people intending to attend the debate what the major points both for and against C and C++ will be, and the lines of arguments that are liable to be used to show how important they are. You are not required to come down either in favour of or against the languages (but may if you wish). [20 marks]

1997 Paper 5 Question 5

Programming in C and C++

A slightly clumsy programmer had been lagging behind the company productivity targets and needed to write some C++ code in a hurry. Almost remembering an old optimising compilers question from student days, this programmer produced a file containing the text:

```
struct List { int head; struct list *tail; };

struct List readlist()
{   int i;
    struct List *p, *q, *t;
L1: p = NULL;
L2: while (scanf("%d", i) = 1)
    /* scanf reads an integer and returns 1
       if it finds one correctly /*
    [
L3:     t = malloc(sizeof(List *));
        if (t == 0) printf("oops no memery\n");
        else t->hd = i;
L4:          t->tl = 0;
        if (p == NULL)
            p = q = t;
        else
            q->tl = t, q = t;
    }
L5: return p;
}
```

Unfortunately the programmer had forgotten what this was supposed to achieve; you are asked to help re-create an explanation for the code and to identify problems (of either style or correctness) in it. You do not need to provide a correct version of the program: just draw attention to as many errors or oddities as you can. Suggest two ways in which a move from C to C++ might allow the structure of the code to be improved. [20 marks]

1997 Paper 6 Question 5

Programming in C and C++

Write a declaration of a C++ class that might be used to implement a binary tree with each node able to hold an integer. Your implementation (i.e. the class itself and those bodies which conveniently fit within it) should make it impossible for casual programmers to access the pointer fields that link parts of the tree together except through cleanly specified access functions. Show how you would overload the "+" operator in C++ to provide a neat notation for adding a new item into such a tree. [20 marks]

2007 Paper 3 Question 4

Programming in C and C++

A C programmer is working with a little-endian machine with 8 bits in a byte and 4 bytes in a word. The compiler supports unaligned access and uses 1, 2 and 4 bytes to store `char`, `short` and `int` respectively. The programmer writes the following definitions (below right) to access values in main memory (below left):

Address	Byte offset			
	0	1	2	3
0x04	10	00	00	00
0x08	61	72	62	33
0x0c	33	00	00	00
0x10	78	0c	00	00
0x14	08	00	00	00
0x18	01	00	4c	03
0x1c	18	00	00	00

```
int **i=(int **)0x04;

short **pps=(short **)0x1c;

struct i2c {
  int i;
  char *c;
}*p=(struct i2c*)0x10;
```

(*a*) Write down the values for the following C expressions:

```
**i          p->c[2]          &(*pps)[1]          ++p->i
```

[8 marks]

(*b*) Explain why the code shown below, when executed, will print the value 420.

```
#include<stdio.h>

#define init_employee(X,Y) {(X),(Y),wage_emp}
typedef struct Employee Em;
struct Employee {int hours,salary;int (*wage)(Em*);};
int wage_emp(Em *ths) {return ths->hours*ths->salary;}

#define init_manager(X,Y,Z) {(X),(Y),wage_man,(Z)}
typedef struct Manager Mn;
struct Manager {int hours,salary;int (*wage)(Mn*);int bonus;};
int wage_man(Mn *ths){return ths->hours*ths->salary+ths->bonus;}

int main(void) {
  Mn m = init_manager(40,10,20);
  Em *e= (Em *) &m;
  printf("%d\n",e->wage(e));
  return 0;
}
```

[4 marks]

(*c*) Rewrite the C code shown in part (*b*) using C++ primitives and give *four* reasons why your C++ solution is better than the C one. [8 marks]

2007 Paper 11 Question 3

Programming in C and C++

A C programmer makes use of the `goto` construct as follows:

```
int test() {
  int x=0,y=0,i,j;
  int err=0;
  if ((y=init())==-1)
    goto error;
  for (i=1;i<10;i++) {
    for (j=1;j<10;j++) {
      if ((x=process(i,j))==-1) {
        err = 10*i+j;
        goto error;
      }
      y += x;
    }
  }
  return y;
 error:
  printf("Something went wrong: %d %d\n",err/10,err%10);
  exit(1);
}
```

(*a*) Rewrite the code in C, maintaining the same functionality but avoiding the use of `goto`. [3 marks]

(*b*) By defining a suitable C++ class to contain the error parameters `i` and `j`, rewrite the above code using C++ exceptions. [5 marks]

(*c*) Write a definition in C *or* C++ for a function `concat` that takes two strings `s1` and `s2` and returns a `char` pointer to heap memory containing a copy of the concatenation of `s1` and `s2`. [5 marks]

(*d*) Write a macro `CONCAT` that takes two string literals as arguments and results in them being concatenated into a single string after the preprocessor has run. [2 marks]

(*e*) Give *two* reasons why the following code is wrong:

```
#define b "UoCCL"
char a[] = "UoCCL";
char i[] = CONCAT(b,a);
char j = concat(a,b);
```

and outline the key differences between `CONCAT` and `concat`. [5 marks]

2008 Paper 3 Question 3

Programming in C and C++

A hardware engineer stores a FIFO queue of bits in an `int` on a platform with 32-bit `int`s and 8-bit `char`s using the following C++ class:

```
class BitQueue {
  int valid_bits; //the number of valid bits held in queue
  int queue;      //least significant bit is most recent bit added
 public:
  BitQueue(): valid_bits(0),queue(0) {}
  void push(int val, int bsize);
  int pop(int bsize);
  int size();
};
```

(*a*) Write an implementation of `BitQueue::size`, which should return the number of bits currently held in `queue`. [1 mark]

(*b*) Write an implementation of `BitQueue::push`, which places the `bsize` least significant bits from `val` onto `queue` and updates `valid_bits`. An exception should be thrown in cases where data would otherwise be lost. [5 marks]

(*c*) Write an implementation of `BitQueue::pop`, which takes `bsize` bits from `queue`, provides them as the `bsize` least significant bits in the return value, and updates `valid_bits`. An exception should be thrown when any requested data is unavailable. [4 marks]

(*d*) The hardware engineer has built a communication device together with a C++ library function `send` to transmit data with the following declaration:

```
void send(char);
```

Use the `BitQueue` class to write a C++ definition for:

```
void sendmsg(const char* msg);
```

Each of the characters in `msg` should be encoded, in index order, using the following binary codes: `'a'`=`0`, `'b'`=`10`, `'c'`=`1100`, and `'d'`=`1101`. All other characters should be ignored. Successive binary codes should be bit-packed together and the code `111` should be used to denote the end of the message. Chunks of 8-bits should be sent using the `send` function and any remaining bits at the end of a message should be padded with zeros. For example, executing `sendmsg("abcd")` should call the `send` function twice, with the binary values `01011001` followed by `10111100`. [10 marks]

2009 Paper 3 Question 1

Programming in C and C++

Explain *five* of the following C or C++ features. You may use a short fragment of code to complement your explanation.

(*a*) The declaration of a C++ class illustrating constructor, variable, and method

(*b*) The use of a virtual destructor

(*c*) The difference between `malloc()` & `free()`
and
`new` & `delete`

(*d*) Overloading an operator

(*e*) Pointer arithmetic

(*f*) Catching and throwing exceptions including the passing of a user-defined structure

(*g*) The meaning of the keywords `static` and `const`

[4 marks each]

2010 Paper 3 Question 6

Programming in C and C++

(*a*) Popular programming journal *Obscure C Techniques for Experts* has published a novel way to save space for a doubly-linked list program. Instead of storing two pointers (one next and one previous), this new technique stores a single value: the `XOR` of *previous* and *next* pointers.

A traditional two-pointer linked list might be illustrated as:

$$\begin{array}{ccccccccccc} \cdots & \text{A} & & \text{B} & & \text{C} & & \text{D} & & \text{E} & \cdots \\ & & \rightarrow & \text{next} & \rightarrow & \text{next} & \rightarrow & \text{next} & \rightarrow & & \\ & & \leftarrow & \text{prev} & \leftarrow & \text{prev} & \leftarrow & \text{prev} & \rightarrow & & \end{array}$$

In contrast, the proposed new technique stores a bit-wise `XOR` of the *previous* and *next* pointers within a single field.

$$\begin{array}{ccccccccccc} \cdots & \text{A} & & \text{B} & & \text{C} & & \text{D} & & \text{E} & \cdots \\ & & \leftrightarrow & \oplus & \leftrightarrow & \oplus & \leftrightarrow & \oplus & \leftrightarrow & & \end{array}$$

You have been engaged to provide code examples of this approach for publication.

Ensure your code illustrates the creation and initialization of such a list as well as the insertion, and deletion, of elements from such a list. Additionally, you must provide examples of a forward or backward traversal of the list permitting examination of each element in turn. [15 marks]

(*b*) Comment on this form of linked list. Consider the comparative speed, memory overheads, maintenance and other advantages or disadvantages of the `XOR` doubly-linked list approach when compared with an approach that stores both previous and next pointers. [5 marks]

2011 Paper 3 Question 3

Programming in C and C++

In this question, where appropriate, you may use a short fragment of code to complement your explanation.

(*a*) What is the difference between declaration and definition?

(*b*) Describe the layout of the memory components: Dynamic Memory Allocation, Data Segment, Code Segment and Stack. You may use an illustration as part of your explanation.

(*c*) Using an example, explain what *stack unwinding* is in C++.

(*d*) How may I use template meta programming to inline recursive functions?

(*e*) Why did the designers of the C++ language decide to make an empty class 1 byte in length?

[4 marks each]

3 Programming in C and C++ (SCC)

In this question, where appropriate, you may use a short fragment of code to complement your explanation.

(*a*) (*i*) What is the difference between a local and global variable in C? (Consider variable scope, storage and initialisation.)

(*ii*) What are the properties of a static member variable in a C++ class?

[4 marks]

(*b*) (*i*) Briefly explain pointer arithmetic in C. Give an example code snippet involving pointers in which it would be *inappropriate* to use pointer arithmetic, and explain why.

(*ii*) Explain how in some respects pointers are equivalent to arrays, and give one respect in which they differ.

[4 marks]

(*c*) Explain why a function might be declared virtual in a C++ superclass.

[4 marks]

(*d*) (*i*) How does use of the void $*$ pointer in C allow a form of polymorphism? Give an example function declaration using the void $*$ pointer.

(*ii*) What is the main problem with the use of void $*$, and how does C++ improve on this? Give the improved function declaration in C++ for your example function in part (*d*)(*i*).

[4 marks]

(*e*) (*i*) Why might it be useful to define a copy constructor for a C++ class? Give an example of a copy constructor for a simple class.

(*ii*) Why might it be useful to explicitly define the assignment operator (=) for a C++ class? Give an example definition of the assignment operator for a simple class.

[4 marks]

3 Programming in C and C++ (AM)

(*a*) You have acquired a C program which declares an array `v`, populates it, and later writes it to a file in binary:

```
#define NITEMS 100
struct Elem { signed long val; char flags; } v[NITEMS];
...
fwrite(file, 1, sizeof(v), v);
```

When run on a legacy processor (which no longer exists) this produces a file containing 500 bytes, but when re-compiled and executed on three modern desktops using various compilers it produces files containing respectively 800, 1200 and 1600 bytes. On all implementations `char` is an 8-bit value.

(*i*) Explain what might be happening in the four versions in terms of compiler assumptions of alignment and size, in bits, of type `long`. Also give the values of `sizeof(struct Elem)`. [3 marks]

(*ii*) You now wish to read a file produced by the legacy processor into a version of the program running on your new desktop machine (one of the three above). Outline the changes, if any, you would need to make to the call to `fread`, mirroring the call to `fwrite` above, so that the resulting `v` may be successfully processed by the rest of the program. You may make any sensible assumptions about the legacy machine and your desktop machine provided you state them explicitly. Indicate how your program might be able to read both legacy- and new-format binary files. [10 marks]

(*b*) (*i*) Write a C++ class `T` which contains a `const` integer field `n`. `T` should also have constructor(s) which initialise `n` to an integer argument passed as a parameter or to zero if no argument is given; `T` should also have a destructor. The constructor(s) and destructor should print the value of the `n` field of the object being constructed or destructed. Indicate why, or why not, any of your fields or methods are qualified with `virtual`. [3 marks]

(*ii*) Explain how objects of class `T` are allocated and deallocated, for each of the three areas: heap, stack and static store, noting one case where appropriate use of `virtual` is essential. What, if any, overlap in programmer convenience is there between stack-allocated objects with destructors and `try-finally` in Java? [4 marks]

COMPUTER SCIENCE TRIPOS Part IB – 2014 – Paper 3

3 Programming in C and C++ (AM)

(*a*) Write a C function `revbits()` which takes a single 8-bit `char` parameter and returns a `char` result by reversing the order of the bits in the `char`. [4 marks]

(*b*) Write a C function `revbytes()` taking two parameters and returning no result. The first parameter is a pointer to memory containing n contiguous bytes (each of type `char`), and the second is the number of bytes. The function should have the side effect of reversing the order of the bits in the n contiguous bytes, seen as a bitstring of length $8n$. For example, the first bit of the first `char` should be swapped with last bit of the last `char`. [6 marks]

(*c*) You have been assigned the following seemingly working C code, which processes files controlling the behaviour of a system. You observe that, after obtaining several `ERR_MALFORMED` errors, subsequent calls to `fopen` fail due to too many files being open:

```
int process_file(char *name)
{   FILE *p = fopen(name, "r");
    if (p == NULL) return ERR_NOTFOUND;
    while (...)
    {   ...
        if (...) return ERR_MALFORMED;
        process_one_option();
        ...
    }
    fclose(p);
    return SUCCESS;
}
```

(*i*) Explain how to fix the program using facilities in C. [2 marks]

(*ii*) Now suppose the function above was part of a system written in C++ (but still using the C file-processing commands such as `fopen` and `fclose`), and that `process_one_option()` might raise one or more exceptions. Using a class with a destructor, show how to fix the "too many files open" bug above. [8 marks]

COMPUTER SCIENCE TRIPOS Part IB – 2015 – Paper 3

1 Programming in C and C++ (AVSM)

A spacecraft arrives at Mars, but its memory has been corrupted by radiation en route. Luckily, it can receive updates one bit at a time using a predefined C function `short receive_bit(void)`, that when called will return either 1 or 0. The stream of bits for a value is transmitted in unsigned big-endian byte order: for example, a 16-bit value of `125` would be `0000000001111101`. Assume the `int` type is 32 bits.

(*a*) Explain the meaning of the `inline` keyword on C function declarations, and a potential drawback of using it. [2 marks]

(*b*) Using `receive_bit()`, define a function `receive_int()` that decodes and returns a 32-bit value from the sequence of received bits. [4 marks]

(*c*) Build a more general decoding function `receive` using a C++ template with two parameters that specify the number of bits to decode and a datatype for the decoded value. Use this to write two template instantiations that decode an 8-bit value into a `short` and a 32-bit value into an `unsigned long`. [6 marks]

(*d*) Find and explain *four* instances of undefined behaviour that could result from compiling and running the C code below with different command-line arguments. The `strcpy(dst,src)` function copies a zero-terminated C string from the `src` buffer to the `dst` buffer. The `putchar(c)` function outputs a character `c` to the console. You can assume that the standard C header prototypes have been included for `<stdio.h>`, `<stdlib.h>` and `<string.h>`. [8 marks]

```
1.      char *show_instruction(int msg) {
2.        char buf[6];
3.        int fuel;
4.        if (msg == 1 && fuel--) {
5.          strcpy(buf, "THRUST");
6.          return buf;
7.        } else if (msg == 2) {
8.          char *msg = (char *)malloc(100);
9.          strcpy(msg, "DEPLOY_PARACHUTE");
10.         return msg;
11.       }
12.     }
13.
14.     int main(int argc, char **argv) {
15.        char *msg;
16.        msg = show_instruction(argc);
17.        putchar(msg[0]);
18.        return 0;
19.     }
```

2 Programming in C and C++ (AVSM)

In this question, you may use short fragments of C or C++ code to complement your answer, where appropriate. Give a brief explanation of the following aspects of C and C++:

(*a*) The differences between C pointers and C++ references. [*Hint:* Consider issues of syntax, initialisation, mutation and safety in your answer.] [5 marks]

(*b*) How C and C++ object files may be safely linked with each other, and the limitations in doing so for some C++ features. [5 marks]

(*c*) The difference between implementation-defined behaviour and unspecified behaviour in the C standard, and an example of each sort of behaviour. Also briefly discuss why these loosely specified behaviours exist in the C standard instead of being strictly defined. [5 marks]

(*d*) The role of a debugger such as LLVM's `lldb` in locating bugs in C code, including the use of breakpoints and watchpoints in an interactive debugger and how symbol tables are useful. [5 marks]

COMPUTER SCIENCE TRIPOS Part IB – 2016 – Paper 3

1 Programming in C and C++ (AVSM)

(*a*) Explain the difference between `'x'` and `"x"` when used as constants in C. Describe the memory representation of both values. [4 marks]

(*b*) Consider the following C program:

```
void swap(int x, int y) {
  int temp = x;
  x = y;
  y = temp;
}

int main(int argc, char **argv) {
  int x = 0;
  int y = 1;
  swap(x, y);
  assert(x == 1);
  return 0;
}
```

Briefly explain the role of the `assert` statement and why this program will trigger an `assert` failure when executed. Supply *two* modified versions of the program that alter the `swap` function definition and, if necessary, its calls, to avoid this `assert` failure. One version should be in C, and the other should use C++ language features. [4 marks]

(*c*) Describe the address-space layout (highlighting four areas of memory) of a typical compiled x86 C program, and how each of these areas are used by C constructs. [8 marks]

(*d*) Briefly explain what *undefined behaviour* is in the C standard. Under what circumstance(s) would calling the following C function result in undefined behaviour?

```
int32_t divide(int32_t a, int32_t b)
{
  return a / b;
}
```

[4 marks]

COMPUTER SCIENCE TRIPOS Part IB – 2016 – Paper 3

2 Programming in C and C++ (AVSM)

(*a*) Consider *unspecified behaviour* in C.

(*i*) Define what unspecified behaviour means in the C standard and give two examples of such behaviour. [3 marks]

(*ii*) Briefly explain why it is important to have unspecified behaviour in the definition of the C language. [1 mark]

(*b*) Compare and contrast the `struct` and `union` keywords in C, supplying an example of a situation where it would be more appropriate to use a `union` rather than a `struct`. [4 marks]

(*c*) Explain the following C or C++ language concepts. You may find it helpful to use short code fragments or diagrams to illustrate your answer.

(*i*) The `virtual` keyword used to qualify a C++ member function and its impact on generated code. [4 marks]

(*ii*) The role of the C preprocessor in the source-code compilation cycle, and why it is a useful tool for debugging. [4 marks]

(*iii*) Templated functions in C++, giving one benefit and one drawback of using them compared with using a `void*` function in C. [4 marks]