UNIVERSITY OF DUBLIN

TRINITY COLLEGE

Faculty of Engineering, Mathematics & Science
School of Computer Science & Statistics

Integrated Computer Science Programme **Trinity Term 2012**

B.A. (Mod.) Business and Computing

Senior Freshman Examination

Systems Programming I and II (CS2014/5)

Wednesday 9th May 2012 **Luce Hall Lower** **14:00 – 17:00**

Dr David Gregg

Instructions to Candidates:

- Answer 4 out of the 6 questions
- All questions are marked out of 25
- All program code should be commented, indented and use good programming style

Materials permitted for this examination:

- Calculator.

1. Bad programming often makes programs difficult to understand, and unfortunately we often have to understand poorly written code. Describe what the following pieces of C code do, and write new versions that provide the same functionality but with simple programming style, appropriate function and variable names, and suitable indentation.

```
/* code section A */
int e (char * a) {char * b = a; while (*(b++)); return b – a – 1;}
```

[5 marks]

```
/* code section B */
int f (int a, int b) {
        return ((unsigned) a) > b;
}
```

[5 marks]

```
/* code section C */
int g (int * a, int b, int c) {
        if ( b == c ) return a[b];
        return g(a, b, (b+c)/2) + g(a, (b+c)/2+1, c);
}
```

[7 marks]

```
/* code section D */
int h (int * x, int y) {
        int * z = x-- + y, w = 0;
        while ( (long) z-- ^ (long) x ) w += !!(*z);
        return w;
}
```

[8 marks]

2. Many applications allocate very large numbers of blocks of memory of the same size. For example, a program that uses a lot of linked lists may allocate large numbers of list nodes. The C functions *malloc* and *free* are designed to deal with memory blocks of any size, and as a result of this generality they incur significant time and space overheads.

   One solution to this problem is to use a memory pool. For example, a memory pool for linked lists of integers might allocate a single piece of memory large enough for 256 linked list nodes using a single call to malloc. These list nodes are then added to a free list within the memory pool abstract data type (ADT). When memory is requested from the ADT, it returns an item from the free list. If the free list is empty, then the ADT must allocate another large block of memory for more list nodes. When a list node is no longer needed, it is returned to the free list. When the memory pool is no longer needed, each of the large blocks of memory must be freed. Write a C ADT representing a memory pool. The ADT should support the following functions:

```
// create a new memory pool with memory for size list nodes
mempool* mempool_new(int size);

// allocate a list node from the memory pool
listnode * mempool_allocate(mempool* pool);

// return a list node to the memory pool
void mempool_free(mempool * pool, listnode* item);

// destructor; free all memory used by memory pool
void listpool_delete(mempool * pool);
```

[25 marks]

3. Some older programming languages, such a PL/1, have types that represent decimal numbers. For example, a currency type might consist of six decimal digits (representing euros), followed by a decimal point, followed by two decimal digits (representing cents). These values were represented internally by an array of decimal digits with a value between 0 and 9, with each array element corresponding to one digit of the number.

   Write a C++ class to represent currency values with six digits before the decimal point, and two after. Your class should provide +, - *, >, <, ==, >= and <= operators. You do not need to provide a division operator. You should also overload the << and >> operators for input and output.

   [25 marks]

4. On most computers the smallest sized piece of memory that can be accessed is a byte. However, for some applications it is useful to be able to operate on nybbles, where a nybble is four bits, or half a byte. When we store arrays of nybbles, ideally we would be able to pack two nybbles into each byte in memory.

   Write a C++ class to represent an array of nybbles. Your class should store the nybbles internally by packing two nybbles into each byte of storage. Your class should have the following prototype:

```
class nybble_array {
private:
        // private data structures
public:
        nybble_array(int size); // create new array of nybbles
        // create new array of nybbles, with each initialized to value
        nybble_array(int size, int value);
        // return the nybble value at position index
        int get_nybble_value(int index);
        // set the nibble at position index to value
        void set_nybble_value(int index, int value);
        ~nybble_array(); // destructor
}
```

5. C++ Standard Template Library (STL) provides a set of standard container classes for use in C++ programs. One of these is the doubly-ended queue class, which implements a queue where values can be pushed or popped from either end. The following shows the broad outline of a simplified version of the STL list class.

```
template <class T>
class mydeque{
private:
        // add your own private variables and methods here
public:
        mydeque( );             // create new, empty deque
        ~mydeque( );            // destructor
        void push_back(T item);     // add new item onto end of deque
        void push_front(T item);    // add a new item to front of deque
        T pop_back( );          // remove and return last item of deque
        T pop_front( );         // remove and return first item of deque
        T& operator[ ](int  i); // return the i'th element of the deque
};
```

Add the remaining declarations to this class, and provide the bodies of methods to implement each of the methods listed above. You may use the basic building blocks of the C++ language (arrays, classes) to construct your class, but you may not use the STL in your code.

[25 marks]

6. (a) The following is a declaration of the type for nodes of a binary tree.

```
struct bintree_node {
        double data;
        struct bintree_node * left;
        struct bintree_node * right;
};
```

Write a C function that takes a pointer to the root of a binary tree and frees the memory occupied by the binary tree. The prototype of your function should be:

```
void free_bintree(struct bintree_node * root);
```

[13 marks]

(b) Write egrep regular expression commands to find all lines in a file described by each of the following:

(i) Lines containing the string 'empty'. [1.5 marks]

(ii) Lines containing any upper case letter. [1.5 marks]

(iii) Lines which end with a question mark. [1.5 marks]

(iv) Lines containing a single letter. [1.5 marks]

(v) Lines containing the same character repeated twice in a row. E.g. "XX". [1.5 marks]

(vi) Lines containing words ending in "our", but excluding the word "our". [1.5 marks]

(vii) Lines containing a date in the format dd/mm/yyyy, e.g. 03/04/1974. [1.5 marks]

(viii) Lines containing the same word more than once. [1.5 marks]