Question 1 (11 marks)

Each of the following descriptions represents a datatype. For each one, using C code:

I) Declare the datatype.
II) Declare one variable having that datatype (not a pointer to it).
III) Initialise the variable. (The actual values are your choice. The simplest possible initialisation code will suffice, but you must completely initialise the variable.)

Choose any valid names, where names have not already been given.

(a) An enum called Rivers with the possible values Rubicon, Thames, Volga, and Nile, having corresponding integer values 3, 4, 6 and 2. **[2 marks]**

(b) A struct containing (1) a two-dimensional array with no fixed size, and (2) a pointer to a function that imports a character, imports a pointer to an integer, and returns a floating point number. **[3 marks]**

(c) A struct named Vertex which is composed of (1) a pointer to a Vertex struct named 'prev', (2) a pointer to a Vertex struct named 'next', (3) an array of strings to be allocated on the heap. **[3 marks]**

(d) A union that can store (1) the enum in part (a), (2) the struct in part (b), or (3) a single real number. **[3 marks]**

Question 2 (14 marks)

Write a C function called halveOrDouble, which imports one pointer to an integer, **a**, and one integer, **n**. The function should not return anything.

The function must first randomly choose whether to perform halving or doubling operations (you may assume a 50% chance for one or another). The function must then perform the chosen operation as many times as indicated by parameter **n**. For example, a call to this function where **a** points to the number 4 and **n** 2 may result in **a** pointing to 1 *OR* 16.

For completeness, you must seed the random number generator in your function.

For this function, you MUST NOT use any looping mechanism!

Hint: Use bitwise operators.

Question 3 appears on the next page

Question 3 (12 marks)

(a) How would you display all of the files in the "UCP" directory and sub-directories that start with either 'prac' or 'Prac', followed by an underscore, ending in two digits, with either the ".c" or ".o" extension.

Hint: use the find command. **[6 marks]**

(b) For the following UNIX shell script, explain the purpose and possible output.

```
for file in $(find); do
    case $file in
        (*$(date + %d)*) echo $file ;;
    esac
done
```

[6 marks]

Question 4 appears on the next page

Question 4 (15 marks)

The following function implements an automated landing feature of an experimental aircraft. When given information about an airport (via an imported pointer a struct), it will steer the aircraft towards the airport and begin descending at a safe speed. The pilot may, however, cancel this behaviour at any time by taking back the controls.

The `autoLand()` function itself is defect-free, but there are defects in the other functions it calls (the code for which is not shown here).

```c
void autoLand(Airport* airport)
{
    float direction = 0.0f;
    float height = 0.0f;
    int* position;

    position = (int*)malloc(2*sizeof(int));

    contactAirport(airport);

    printf("Beginning automated landing.\n");

    /* Make sure the pilot is not trying to take back control.*/
    while(notPiloted())
    {
        getPosition(position);
        direction = getDirection();
        height = getHeight();

        if(direction != airportDirection(airport))
        {
            printf("Steering towards airport.\n");
            adjustDirection();
        }
        else if(height > 0)
        {
            descend();
        }

    }
    free(airport);
    free(position);
    printf("Success!\n");
}
```

Question 4 continues on the next page

For each situation below:

I) Give two plausible hypotheses for what might be wrong (in the code not shown).
II) How do the hypotheses fit the observations?
III) What debugging steps (e.g. breakpoints, monitoring of particular variables) will help narrow down the problem?
IV) How will you know which hypothesis is correct (or more likely to be correct)?

(a) Even if the aircraft is known to begin in the correct direction, it always turns left. **[5 marks]**

(b) Regardless of height, the aircraft never descends. **[5 marks]**

(c) A segmentation fault occurs after the program displays "Beginning automated landing.". **[5 marks]**

Question 5 appears on the next page

Question 5 (48 marks)

For this question, refer to the following standard C function prototypes:

```
FILE *fopen(const char *path, const char *mode);
int fclose(FILE *fp);
int fscanf(FILE *stream, const char *format, ...);
char *fgets(char *s, int size, FILE *stream);
int feof(FILE *stream);
int strcmp(const char *s1, const char *s2);
char *strncpy(char *dest, const char *src, size_t n);
int atoi(const char *nptr);
```

(a) Declare suitable C datatypes to represent each of the following sets of information (as you would do in a header file):

(i) A student. Their name may be up to 100 characters long, their ID up to 8 characters long (as it may be made up of both numbers and letters), and an integer that represents their date of birth. **[2 marks]**

(ii) A class of students of unknown size. The class must also contain a unit code as a string made up of up to 8 characters.
Hint: Marks have been allocated for using a linked list. **[6 marks]**

(iii) A structure representing a (small) University. The University should be separated by a series of classes in Engineering and a series of classes in Computing. **[5 marks]**

Question 5 continues on the next page

(b) Write a C function called `loadClasses` to read a list of classes and their students from both Computing and Engineering.

The file will be divided into two parts: all the Computing classes, then all the Engineering classes. Additionally, the file will be formatted to first indicate the number of classes in Computing and the number of students in each class (you may assume each class has the same number of students). The unit code for the first class will follow on the next line. The next *three* lines will represent the first student's name, then ID, then date of birth in that order. Each subsequent set of three represents another student until there are no more to be listed. The pattern will repeat once more from the next class' unit code.

After all Computing classes have been listed, there will be a separator: `---`. The lines after these three hyphens contain all Engineering classes. The format for these will be the same as before.

The following is an example file:

```
2 1
COMP1000
Arlen Brower
1234567A
01011980
COMP2008
Antoni Liang
7654321B
02021980
---
1 2
ELEN1000
Antoni Liang
7654321B
02021980
Felicia Codeman
1231234C
21082010
```

In this example, two Computing classes each with one student and one Engineering class with two students.

Your function should:

- Take in a filename parameter.
- Read the file, according to the above specifications.
- Dynamically allocate the structures you designed in part (a).
- Store the file data in these structures.
- Return a pointer to the main structure from part (a)(iii).

If the file cannot be opened, your function must return NULL. An error message is not required. If the file can be opened, you may assume that it definitely conforms to the specification. **[16 marks]**

(c) Write a C function called `doubleDiscipline` that creates a list of students taking both Computing and Engineering classes based on the lists generated in the previous part. To facilitate this, the function will import a pointer to another function that will perform the comparisons between two students. It will then return true (1) if they are the same student, and false (0) if they are different.

You **DO NOT** need to write the function that this pointer references!

The function `doubleDiscipline` must must:

- Import a pointer to the comparison function. This function returns an integer and imports two Student structs.
- Import a University struct (from the previous part of the question).
- For each student in a Computing class, use the comparison function pointer to determine if the student also appears in an Engineering class:
  - If a match is found, add that student to a list.
  - If no match is found in any of the Engineering classes, the function should move to the next student.
- Once you have assembled the student list, output the name of each student to the user. The list should appear with one student per line.

Hint: use the structure from part a(ii) to store the student list.

Given the example file shown in part (b), your function should output this:

```
The following students are in both Computing and Engineering classes:
    Antoni Liang
```

[14 marks]

(d) Write a `main` function that accepts (and requires) *one or more* filenames as command-line parameters.

Your `main` should:

- Conduct all appropriate error checking and handling.
- Perform all necessary cleaning up.
- For each filename:
  - Use the function `loadClasses` from part (b) to read the file.
  - Use the function `doubleDiscipline` from part (c) to generate the list of students in Engineering and Computing classes.

[5 marks]

End of Examination