Question 1 (11 marks)

Briefly describe each of the following declarations:

(a) `const int* const cat;` [2 marks]

(b) [2 marks]
```
struct Animal {
        char* species;
        int legs;
        int eyes;
    };
```

(c) `typedef enum {DOLPHIN, PORPOISE = 3, WHALE} SeaMammal;` [2 marks]

(d) [2 marks]
```
union Organism {
        Animal a;
        Plant p;
    };
```

(e) [3 marks]
```
typedef struct Herbivore {
        struct Herbivore* related;
        Animal* a;
    } Herbivore;
```

Question 2 (9 marks)

Write one short C program that:

- accepts two command-line parameters;
- converts the two parameters to integers; and
- outputs a random integer within the range given by the parameters.

For instance, your program may be called as follows:

```
./program 5 10
```

In this example, your program should output a single random integer between 5 and 10, inclusive.

Question 3 appears on the next page

Question 3 (20 marks)

Consider the following code.

```
int x[] = {1, 2, 3, 4, 5};
int y[] = {-1, -2, -3, -4, -5};

int* a = &x[1];
int* b = y + x[2];

int** c = (int**)malloc(2 * sizeof(int*));
*c = a;
*(c + 1) = b;

(*c)[0] = *c[1];
*(*c + 1) = c[1][1];
```

Based on this:

(a) Draw a diagram showing all the pointer relationships created. [16 marks]

(b) Show the contents of `a` and `b` at the end. [4 marks]

Question 4 (20 marks)

The following code (shown on the next page) is the `main()` function for a text search program. The program asks the user for a search term and a filename, and then finds and reports all occurrences of the search term inside the specified file.

However, the program has defects! The defects are **not in the code shown here**, but rather in the functions called by `main()`.

Question 4 continues on the next page

```
1  int main() {
2      char* searchTerm;
3      char* filename;
4      char* fileContents;
5      int i;
6      int lastPosition;
7
8      searchTerm = readLine();
9      filename = readLine();
10
11     fileContents = readFile(filename);
12     lastPosition = calcLastPosition(fileContents, searchTerm);
13
14     for(i = 0; i <= lastPosition; i++) {
15         if(compare(searchTerm, fileContents, i)) {
16             printf("Match found at position %d\n", i);
17         }
18     }
19
20     free(searchTerm);
21     free(filename);
22     free(fileContents);
23     return 0;
24 }
```

You are using a debugger (any debugger) to find the defects. For each situation below, describe:

- Where you would place a breakpoint, and why.
- What values/variables (if any) you would monitor, and why.
- Any assumptions you make about relevant functions.

(a) A segmentation fault occurs after the user enters a search term. [5 marks]

(b) The program appears to work correctly, but causes a segmentation fault at the end. [5 marks]

(c) The program ends without finding any matches, even though the file does contain the search term. [5 marks]

(d) The program erroneously "finds" a match at every single position in the file, *except* where a match actually occurs. [5 marks]

Question 5 appears on the next page

Question 5 (40 marks)

(a) Design suitable structures to represent each of the following sets of information. Implement your design in C using `typedef` declarations (as they would appear in a header file):

(i) A test subject, described by

- An identifying letter.
- A test score (a real number).
- A ranking (an integer).

(ii) A collection of test subjects, described by:

- An array of the data type described in part (i).
- The number of elements in the array.

[8 marks]

(b) Write a C function called `readData`, which:

- Imports a filename as a `char` pointer — the input file. This is a text file, structured as follows:
  - The first line contains a single integer — the number of records in the file.
  - Each subsequent line contains one record, consisting of an ID (a character), a test score (a real number) and a ranking (an integer), separated by spaces.

  For example:

```
3
A 5.5 2
B 8.6 1
C 3.4 3
```

- Opens the file for reading.
- Dynamically allocates the appropriate memory for the required data structures from part (a).
- Reads the data from the file into the data structures.
- Returns a pointer to the data structure described in part (a) (ii).
- Returns `NULL` instead if any errors occur, and outputs an appropriate error message.

Ensure that your C code conforms to the characteristics emphasised in the lectures and practical sessions. [15 marks]

Question 5 continues on the next page

(c) Write a C function called `writeResults`, which:

- Imports:
  - A filename as a `char` pointer — the output file.
  - A pointer of the same type returned by the `readData` function from part (b).
- Cycles through all the test subjects to determine the **mean** (average) and **maximum** test score.
- Opens the specified text file for writing.
- Writes the mean and maximum test score to the file on a single line. Both values should be output with 2 decimal places and a field width of 9. For example:

```
     5.83     8.60
```

Ensure that your C code conforms to the characteristics emphasised in the lectures and practical sessions. [10 marks]

(d) Write a `main` function in C which:

- Reads two filenames from the user — the input and output files.
- Uses the function `readData` from part (b) to read the input file.
- Uses the function `writeResults` from part (c) to write the results to the output file.

Ensure that your C code conforms to the characteristics emphasised in the lectures and practical sessions. [5 marks]

(e) Write the appropriate function prototype declarations (as they would appear in a header file) for the functions from parts (b) and (c). [2 marks]

—— End of Supplementary Examination Paper ——