Question 1 (11 marks)

Briefly describe each of the following declarations:

(a) `const int LENGTH = 5;` [1 mark]

(b) `const int* oats;` [1 mark]

(c) [2 marks]

```
struct Complex {
    double real;
    double imaginary;
};
```

(d) `enum Dir {NORTH, EAST, SOUTH = 4, WEST};` [2 marks]

(e) [2 marks]

```
typedef union {
    double orange;
    char lime;
} Jelly;
```

(f) [3 marks]

```
typedef struct Misc {
    FILE* f;
    struct Misc* next;
} Misc;
```

Question 2 (9 marks)

Write a short C program to output a random command-line parameter.

For instance, your program may be called as follows:

```
./randomparam alpha bravo charlie delta echo
```

In this example, your program should output “`alpha`” *or* “`bravo`” *or* “`charlie`” *or* “`delta`” *or* “`echo`”, selected at random with equal probability.

Question 3 appears on the next page

Question 3 (20 marks)

Consider the following code.

```
int a[] = {0, 2, 4, 6, 8, 10};
int b[] = {1, 3, 5, 7, 9};

int* x = NULL;
int* y = NULL;
int** z = NULL;

x = a + a[a[1]];
y = &b[2];
z = (int**)malloc(2 * sizeof(int*));

*z = x;
*(z + 1) = y;

z[0][0] = z[1][0];
z[0][1] = z[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 (on the next page) is the `main()` function for an anagram solver. (An anagram is a word formed by re-arranging the letters of another word.) The program loads a complete English dictionary, prompts the user for a word, and then finds all anagrams of that word (i.e. all other words containing the same letters).

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

The program implements a `Dictionary` ADT (abstract data type), which loads all English words and allows the program to access them one-by-one.

Question 4 continues on the next page

```
int main() {
    Dictionary* dict;
    char* userWord;
    char* dictWord;
    int numWords;
    int i;

    dict = Dictionary_constructor();

    /* Read a word from the user; store it in a dynamically-
       allocated string. */
    userWord = readWord();

    numWords = Dictionary_size(dict);
    for(i = 0; i < numWords; i++) {
        dictWord = Dictionary_getWord(dict, i);

        /* Test if dictWord is an anagram of userWord */
        if(anagramCompare(userWord, dictWord)) {
            printf("%s\n", dictWord);
        }
    }

    Dictionary_destructor(dict);
    free(userWord);

    return 0;
}
```

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 immediately after the user enters a word. [5 marks]

(b) A segmentation fault occurs immediately after the first anagram is output. [5 marks]

(c) The program finishes very quickly, but never finds any anagrams. [5 marks]

(d) The program outputs all the words that are *not* anagrams. [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 identification number (a positive integer).
- Their height (a positive real number).
- Their weight (a positive real number).

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

- An array of the abstract 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 (an integer), a weight (a real number) and a height (a real number), separated by spaces.

  For example:

```
3
45 77.8 166.24
23 65.1 170.9
10 105.51 175.3
```

- 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 minimum height and weight.
- Opens the specified text file for writing.
- Writes the minimum height and weight to the file on a single line. Both values should be output with 4 decimal places and a field width of 10. For example:

```
   65.1000   162.2400
```

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 Examination Paper ——