CURTIN UNIVERSITY
Department of Computing

Test 2 – Semester 2, 2016

SUBJECT: COMP1000 - UNIX AND C PROGRAMMING

TIME ALLOWED:
90 minutes

GENERAL INSTRUCTIONS:

This is a CLOSED BOOK test. Put away all books, notes, etc. Turn off and put away all electronic devices, including calculators and watches. (Make sure your phone is switched off.)

You must write in blue or black pen – pencil will not be accepted.

There are FIVE (5) questions. Write your answers on the question paper. Use the back of the pages if needed (especially for working out).

You have 90 minutes to complete the test. There are 100 marks available.

ATTEMPT ALL QUESTIONS

Name: ____

Student No: ____

Markers Use only – do not write in these boxes

Q1	Q 2	Q 3	Q 4	Q 5	Total /100

Question 1 (10 marks)

(a) Write a suitable set of type declarations for representing a list of football matches, where each match records the names of two competing teams and their integer scores. All matches are played in the same year, and this year should also be recorded. There are no limits on the number of matches, or on the length of team names.

[4 marks]

(b) Based on your answer for part (a), show how to dynamically allocate the necessary memory for a set of 25 matches, where all the team names are at most 15 characters long. Initialisation is not required, except to allocate memory.

[6 marks]

Question 2 (20 marks)

Draw complete, labelled diagrams of memory and/or pointer relationships for the below code segment. The datatype float occupies 4 bytes of memory. Additionally, show the values of all non-pointer variables.

Hint: You have been given an example of two different methods to do this. Either will suffice.

```
float* w = (float*)malloc(3 * sizeof(float));
float* x = (float*)malloc(3 * sizeof(float));
float** y = (float**)malloc(2 * sizeof(float*));
float** z = (float**)malloc(2 * sizeof(float*));
int i;

for(i = 0; i < 3; i++)
{
   *(w + i) = i+4;
   x[i] = 5 - i;
}

*(y + 0) = &x[(int)w[2]];
*(y + 1) = w + (int)x[2];
y[0]-=4;
y[1]-=2;
z[0] = y[1] + 1;
z[1] = x;
z[1][1] += y[1][1];
(*y)[0] += (*z)[0];
```

(Use the back of the previous page for working out)

Question 3 (15 marks)

Write a C function (not a whole program) called `min2D`. The function should:

- Import a fixed-size 2D array of real numbers, with HEIGHT rows and WIDTH columns (where HEIGHT and WIDTH are preprocessor constants already defined elsewhere).
- Also import a fixed-size 1D array of real numbers, with HEIGHT elements.
- For each row in the 2D array, determine the minimum value.
- Export the minimum values using the 1D array.
- Return nothing.

Question 4 (40 marks)

Refer to the following standard C function prototypes (you may not need all of them):

[6 marks]

```
FILE *fopen(char *path, char *mode);
int fclose(FILE *fp);
int fscanf(FILE *stream, char *format, ...);
char *fgets(char *s, int size, FILE *stream);
int fgetc(FILE *stream);
int ferror(FILE *stream);
size_t strlen(const char *s);
```

Also refer to the following declaration:

```
typedef struct
{
   char search;
   char *text;
} Set;
```

(a) Write a function called `readFile` to:

- Import a filename as a parameter.
- Read the contents of the file into a dynamically-allocated array of Set structs. The file is formatted as follows:
  - The first line contains two integers: the number of subsequent lines, and the maximum text length (see below). These values have no fixed limits.
  - Each subsequent line consists of a single non-space character, followed by a space, followed by one or more characters of free-form text. The length of the free-form text will be, at most, the maximum given on the first line.
- For example:

```
4 8
x abc
e elephant
3 1a2b3c
% !@ #$%
```

- Return the array of Sets, and export the array length via a parameter passed by reference.
- If an error occurs, return NULL instead; no error message is needed. You may assume that, if the file exists, it can definitely be read and will be in the correct format.

[20 marks]

(b) Write a function called `countChars` to:

- Import an array of Set structs, and the array length; and return nothing.
- For each Set:
  - Count the number of occurrences of the search character within the word string. (Count exact matches only. Uppercase and lowercase letters are different.)
  - Calculate the proportion (i.e. a real number between 0 and 1) of the characters in word that match search.
  - Output the count and proportion on a single line, separated by a space. The proportion should be expressed with 3 decimal places.

  For example:

  ```
  0 0.000
  2 0.250
  1 0.167
  1 0.143
  ```

  (This is the expected output given the example input file shown previously.)
- Return nothing.

[10 marks]

(c) Write a main function to:

- Take one or more filenames on the command-line.
- For each filename, read the file with `readFile` and, as appropriate, count matching characters with `countChars`.
- Clean up any allocated memory.
- Output any appropriate error messages.

[10 marks]

Question 5 (15 marks)

Consider the following declarations:

```
typedef struct Node
{
   struct Node *next;
   char *text;
} Node;

typedef struct
{
   Node *start;
   Node *end;
} List;
```

Write a function called `listCase` to determine (1) the total number of lowercase characters, and (2) the total number of uppercase characters in the text stored in the list. Do not break this down per node; rather, count all lowercase and uppercase characters across all nodes.

Your function should return nothing, and take three parameters:

- A pointer to List.
- Two pointers to integers, called lower and upper, to export the results.

There should be no input or output, and your function should not modify the list.

END OF TEST