Study Period 4

University	***RMIT University***
Exam Code(s):	**CPT220**
Title of Paper:	***Programming in C***
Exam Duration:	3 hours
Reading Time:	**10 minutes**

During an exam, you must not have in your possession, a book, notes, paper, electronic device(s), calculator, pencil case, mobile phone, smart watch/device or other material/item which has not been authorised for the exam or specifically permitted as noted below.. Any material or item on your desk, chair or person will be deemed to be in your possession. You are reminded that possession of unauthorised materials in an exam is a discipline offence.

All questions in this exam have gone through some basic testing with gcc. This exam is made up of 180 marks which amounts to 1 mark per minute of maximum time you should spend on the question).

No examination papers are to be removed from the room.

Authorised Materials

Calculators	☐ Yes	☒ No
Open Book	☐ Yes	☒ No
Specifically Permitted Items	☐ Yes	☒ No

If yes, specifically permitted items are:

Students must complete this section if required to write answers within this paper

OUA ID:
Provider University ID:
Family name:
Other names (in full):

For each question in this exam other than question 1 you can create any additional functions or data structures of your own if you wish to but you do not have to. Also, you may assume that a BOOLEAN type exists with valid values of FALSE (0) and TRUE (1), even if this is not stated in the question. You may also assume any provided code is correct unless we indication otherwise.

Question 1: Short Answer Questions (7 x 5 marks each = 35 marks)

Answer the following short answer questions in your exam script book. Please do not start each question in this section on a new page as they are all rather short answers.

a) What will the following program print? Please indicate clearly any newline or other whitespace characters. If they are not indicated, they are not part of the output you intended:

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
        char * strings[] =
        {
                "Power", "corrupts", "and", "absolute", "power",
                "corrupts", "absolutely"
        };
        printf("%s\n", strings[3]);
        puts(*(strings+6));
        puts(*(strings+4)+3);
        putchar(*(*(strings+3)+4));
        putchar('\n');
        putchar(*(*(strings+1)+3)+4);
        putchar('\n');
        return EXIT_SUCCESS;
}
```

b) given the follow data structures:

```
struct list_node
{
        int data;
        struct list_node * next;
};
struct list
{
        struct list_node * head;
        size_t num_nodes;
};
typedef enum
{
        FALSE, TRUE
} BOOLEAN;
```

c) Why are the contents of the list header not properly initialized with the following code:

```
BOOLEAN list_init(struct list * list)
{
        list = malloc(sizeof(struct list));
        if(list == NULL)
        {
                return FALSE;
        }
        list->head = NULL;
        list->num_nodes=0;
        return TRUE;
}
```

d) Why does the following C macro behave incorrectly and how do we solve this?

```
#define MULT(X) X*X
```

e) Why does gcc issue a compiler warning when compiling this program? Also, what is the output of the program?

```
int main(void)
{
    enum {
        FIRST, SECOND, THIRD, FOURTH, FIFTH
    } num_times;

    for( num_times = FIRST; num_times <= FIFTH;
           ++num_times)
    {
        switch(num_times)
        {
                case FIRST:
                   printf(" ## just once ##\n");
                        break;
                case SECOND:
                    printf(" ## So here we are again"
                           " ##\n");
                        break;
                case FOURTH:
                    printf(" ## Here for the fourth "
                           "time ##\n");
                case FIFTH:
                    printf(" ## All done now ##\n");
            }
        }
    return EXIT_SUCCESS;
}
```

f) The following is output from valgrind. Please provide as much detail as you can on the cause of this error message.

```
==21416== Use of uninitialised value of size 8
==21416==    at 0x400572: main (1e.c:7)
==21416==
==21416== Invalid read of size 4
==21416==    at 0x400572: main (1e.c:7)
==21416==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
```

g) Consider the following Makefile:

```
TARGETS=first.c second.c third.c
OBJECTS=first.o second.o third.o
HEADERS=first.h second.h third.h shared.h
CFLAGS=-ansi -Wall -pedantic
LFLAGS=

all: project

project: $(OBJECTS)
     $(CC) $(LFLAGS) -o project

%.c:%o $(HEADERS)
     $(CC) $(CFLAGS) -c $<
```

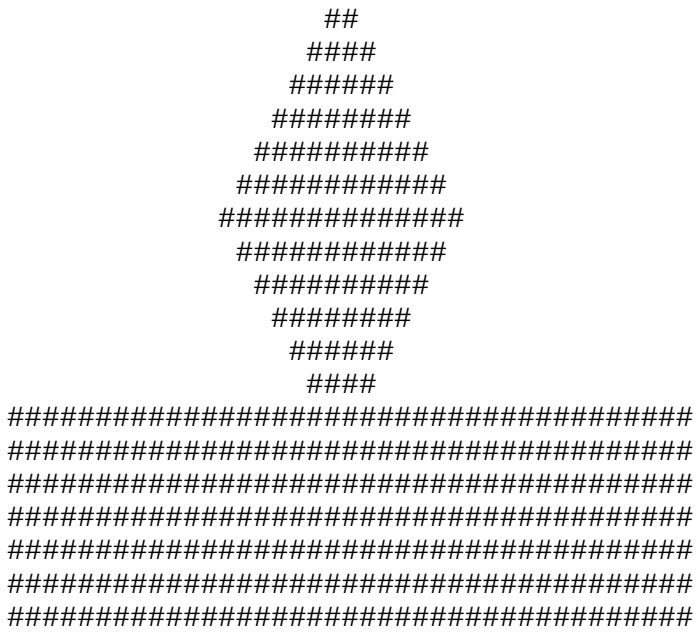
Write a "clean" target for this Makefile that deletes the executable and any object files created by this Makefile.

Question 2: Edge Finding in an ascii image (10 + 25 = 35 marks)

For this question you may assume the following type definitions and constants:

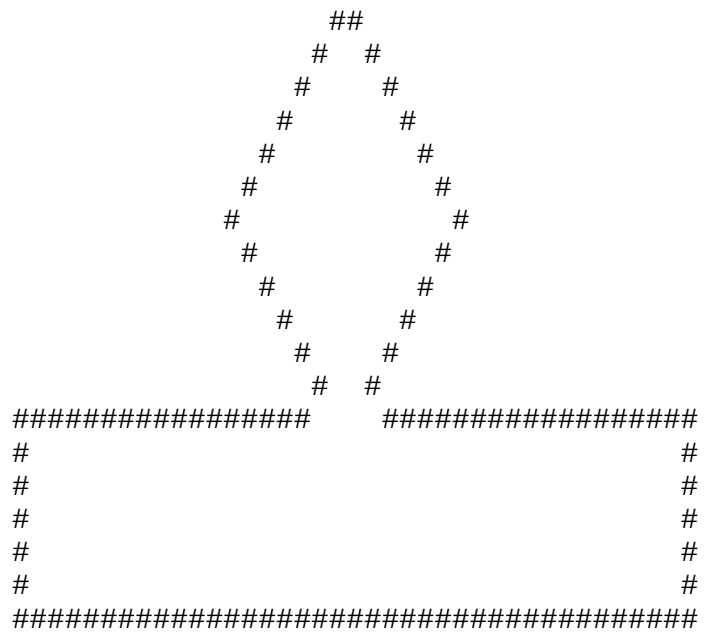
```
#define MAX_PICTURE_HEIGHT 80
#define MAX_PICTURE_WIDTH 80
#define EXTRA_CHARS 2
#define SPACE ' '
#define SOURCE_ARG 1
#define DEST_ARG 2
#define NUM_ARGS 3
typedef enum
{
     FALSE, TRUE
} BOOLEAN;
```

For this question you are processing a text file which represents an image composed of ascii characters such as the following:



2a) Your first task here is to write a function to find the edges in a piece of ascii art. An edge is defined as a non-space character which has space characters for neighbours. You should copy a character to the destination array only if it is an edge character so defined. You may assume that the destination array starts off as totally containing space characters (as output by the space bar) but the contents of this array is NOT strings but it is an array of arrays of char - they are not nul terminated.

For example, the contents of the destination array after having processed the above image is:



The function prototype for this question is:

```
void find_edges(
     char source_image[MAX_PICTURE_HEIGHT][MAX_PICTURE_WIDTH],
     char dest_image[MAX_PICTURE_HEIGHT][MAX_PICTURE_WIDTH]);
```

2b) Write a COMPLETE PROGRAM that makes use of the function you have designed above.

The program should utilise command line arguments to allow the caller of the program to specify two files to open - the first contains a piece of ascii art surrounded by whitespace characters and the second may or may not exist yet.

You are to check for correct command line arguments and do appropriate initialization of arrays and other data structures (arrays should be initialized to contain all spaces and not be nul terminated).

You should then open the first file and load it into the source 2d array. You should then call the above defined function correctly. After return from the above function, open the destination file and save the edge-detected version of the loaded image.

Question 3: Tokenization and String Processing (15 + 25 = 40 marks)

For this question you assume you have available the following constants and data structures:

```
#define MAXWORDS 10
#define LINE_LEN 80
#define EXTRA_CHARS 2
#define NUM_ARGS 2
#define FILE_ARG 1
typedef enum
{
     FALSE, TRUE
} BOOLEAN;
```

3a) For this function you will write a function to accept a string as a parameter along with an array of char pointers in which to store tokens received and a string containing the delimiters. You should make a copy of the line passed in and then tokenize this line according to the delimiters passed in. Each token should be stored in the tokens array. You should make every effort to perform correct memory management (allocating and freeing of memory).

The function prototype of this function is:

```
BOOLEAN tokenize(const char * line, char * tokens[MAXWORDS],
                 const char * delimiters);
```

3b) Write a COMPLETE PROGRAM that allows the user to specify a file to find the longest word in.

It opens a text file for reading and then tokenizes each line using the above defined function.

It then iterates over each line token of each line checking for a word longer than the ones it has seen so far. When it finds it, that becomes the new longest word.

It should print out the longest word at the end. You should perform basic validation as part of this function and do your best to do appropriate memory and resource (files, etc) management.

Question 4: 2d Arrays, random numbers and pointers (20 marks)

For this question, you may find the following data structures and constants from assignment 1 will come in handy:

```
typedef enum
{
     FALSE, TRUE
} BOOLEAN;
#define BOARDHEIGHT 3
#define BOARDWIDTH 3
enum cell
{
        C_NOUGHT='0',
     C_CROSS='X',
     C_EMPTY=' ',
     C_INVALID=-1
};
```

We have decided to ask you to implement an extension of assignment 1. You are to implement the following function:

```
BOOLEAN swap_token( enum cell board[BOARDHEIGHT][BOARDWIDTH],
          const unsigned oldx, const unsigned oldy,
          enum cell expected, unsigned * newx, unsigned *newy);
```

Where board is the tictactoe game board, oldx and oldy are the old (current) coordinates for a token and expected is the token we expect to find there.

Your first task is to check whether the token at that location is what is expected and if not return FALSE.

If the token is what was expected, you need generate random x and y cooordinates selecting potential spots to move the token to until the coordinates refer to an empty spot on the board.

At this point you want to assign an empty space to the old coordinates and assign the token that was there to the new coordinates. You then want to return those new coordinates via the pointers newx and newy and return TRUE.

Question 5: Linked Lists, Assignment 3 Extension (5+20+25 = 50 marks)

Consider the following data structures defined for assignment 3:

```
struct word
{
     char * word;
     size_t count;
     int cost;
};
struct list_node
{
     struct word * data;
     struct list_node * next;
};

struct word_list
{
     struct list_node * head;
     size_t num_words;
     int overall_cost;
};

typedef enum
{
     FALSE, TRUE
} BOOLEAN;
```

The company that commissioned your assignment 3 project has decided that it is going to create a publishing system as a separate module however it needs you to write some extensions to your assignment 3 project that will allow it to do some costings.

In the new publishing system it is going to charge a publishing cost for each post. Each word 5 characters or shorter is going to cost 5 cents, words between 5 and 10 characters will cost 10 cents and anything longer will cost 20 cents.

5a) Software Design:

Your task in this first section is to alter the provided data structures so we can keep track of the cost of each word (that is the cost for one occurrence of the word times the number of times it has appeared) as well as the overall cost for all words currently in the collection.

5b) Calculate Costs:

Your task in this section is to iterate over the word list for each word and calculate its cost in cents and assign it to the cost variable you created for each word; you should then assign the total cost in cents to the variable you placed in the the header structure for the list. The function requirement for this requirement is:

```
void calculate_costs(struct word_list * list);
```

5c) Update to list deletion:

You now need to rewrite the delete function for this list. In this case the delete is pretty much the same as what you implemented in your assignment except that you need to deduct the total cost of the word to be deleted from current overall cost. You should make every effort to keep all other variables up to date, including freeing of memory and returning an appropriate success or failure value. The function prototype for this question is:

```
BOOLEAN list_delete(struct word_list * list, const char * word);
```