Question 1. [5 MARKS]

Part (a) [1 MARK]

The following two C program files, whose content can be seen below, are stored in the current directory.

main.c:

```
#include <stdio.h>

int main(void) {
    int val = get_number();
    printf("%d", val);
    return 0;
}
```

get_number.c:

```
#include <stdlib.h>

int get_number() {
    return rand() % 100;
}
```

The following command tries to create an executable program

```
gcc -Wall -o get_number main.c get_number.c
```

but returns with the following error:

```
main.c: In function main:
main.c:4:15: warning: implicit declaration of function get_number [-Wimplicit-function-declaratio
     int val = get_number();
               ^
```

Explain what should be done to resolve this problem, without modifying the compilation command.

SOLUTION: Add the function's prototype inside the main.c file.

Add the prototype of get_number() in to main.c

Part (b) [1 MARK]

Assume you have a terminal open, and that the parent directory contains a C executable file called my_prog. Write a single command that invokes my_prog passing CSC209 as a command-line argument, also redirecting the program's standard input from a file called values.txt.

SOLUTION: `../my_prog CSC209 < values.txt`

../my-prog CSC209 < value.txt

Part (c) [2 MARKS]

Here is the output from running `ls -l` on the current directory.

number of links to the file owner group size

```
-r-xrw---- 1 craig instrs 157 Aug 13 10:53 prog
```

Explain what you know about the permissions on the prog file. Be specific about who is allowed to do which actions.

SOLUTION: craig can read and execute, members of the instrs group can read and write, others have no permissions at all.

user, craig, can do read, execute, group ,instrs, can read, write, and no other's permission

Part (d) [1 MARK]

Show the output if you were to run these two commands on the current directory.

```
$ chmod 643 prog
$ ls -l prog
```

1 1 0   1 0 0   _ 1 1

-rw-r---wx 1 craig instrs 157 Aug 13 10:53 prog.

SOLUTION:

```
-rw-r---wx 1 user instrs 157 Aug 13 10:53 prog
```

Question 2. [5 MARKS]

Consider the following pieces of code. Fill the tables below with the values of the array elements at the point in the execution where the table appears. The first table is done for you.

% Pointer arithmetics.
% Dereference of pointer affects the pointed value and not the pointer itself.
% Compound assignment operators.
% Modulo operator.

```
int arr[4] = {8, 4, 6, 13};
```

arr[0]	arr[1]	arr[2]	arr[3]
8	4	6	13

8 4 6 13

```
int step = 1;
int *ptr;

ptr = &arr[2];
arr[0] = *ptr + 3;
```

arr[0]	arr[1]	arr[2]	arr[3]
9	4	6	13

9 4 6 13

```
ptr = ptr - step;
*ptr = 10 % 3;
```

arr[0]	arr[1]	arr[2]	arr[3]
9	1	6	13

9 1 6 13

```
*(ptr + 2) = step * 2;
```

arr[0]	arr[1]	arr[2]	arr[3]
9	1	6	2

9 1 6 2

Question 3. [8 MARKS]

Fill in the memory diagram to show the current state of the program exactly before the return statement on **line 11** is executed. Then fill in the blanks in the two sentences at the bottom of the page. For your picture, please assume that integers are 4 bytes and pointers are 8. Label the stack frames.

```
1   char *get_name(char *p) {
2       int i = strlen(p);
3       while (i >= 0 && p[i] != '/')
4           i--;
5
6       i++;
7       char *e_name = malloc(strlen(&p[i]) + 1);
8       strcpy(e_name, &p[i]);
9       p[i] = '\0';
10
11      return e_name;
12  }
13
14  int main()  {
15      char path[] = "tmp/file1";
16      char *name = get_name(path);
17      printf("%s\n", name);
18      free(name);
19      return 0;
20  }
```

Section	Address	Value	Label
Read-only	0x100	tmp/	
	0x104	file	
	0x108	1\0	
	0x10c		
	0x110		
	0x114		
	0x118		
	0x11c		
	⋮	⋮	
Heap	0x23c	f i l p file	
	0x240	1 \0 1\0	
	0x244		
	0x248		
	0x24c		
	0x250		
	0x254		
	0x258		
	0x25c		
	⋮	⋮	
	0x454		
	0x458		
	0x45c		
get_name	0x460	0x23c 0x23c	e_name e_name
	0x464		
	0x468	4 4	i i
	0x46c	0x47c 0x47c	p p
	0x470		
main	0x474	??? ????	name name
	0x478		
path	0x47c	t m p / tmp/	path[0] path[0]
	0x480	\0 i l e \0ile	
	0x484	1 \0 1\0	

The value of variable name **before** the statement on line 18 is executed is 0x23c 0x23c

The value of variable name **after** the statement on line 18 is executed is 0x23c 0x23c

Question 4. [4 MARKS]

The code fragements in this question use the same struct declaration and function definition shown in the box. For each fragment, indicate whether the code works as intended or there is an error. Assume all programs are compiled using the *gnu*99 standard. If the code runs without error, give the output. If there is an error in a fragment, explain **briefly** what is wrong. We have intentionally omitted the error checking of the system calls to simplify the examples. Do **not** report this as an error.

```
struct Car {
    char *colour;
    int mileage;
};
```

```
struct Car create_car() {
    struct Car c;
    c.colour = "Green";
    c.mileage = 1000;
    return c;
}
```

Part (a) [2 MARKS]

```
struct Car car;
car = create_car();
printf("(%s, %d)\n", car.colour, car.mileage);
X strcpy(car.colour, "Orange");
printf("(%s, %d)\n", car.colour, car.mileage);
```

□ Works as intended ☑ Error

cannot assign to an immultable string literal String literal is not multable.

Part (b) [2 MARKS]

```
Question dropped from test because of significant typos.
```

Question 5. [8 MARKS]

Complete the following program to match the specifications provided in the comments.

```
struct node {
    char name[20];
    int assignment;
    char *status; // one of "no sub", "submitted", "marked", "released"
    float grade;
    struct node *next;
};
// Create and return node with name n for assignment a, with status "no sub"
struct node *create_student(char *n, int a) {
    struct node *result = malloc(sizeof(struct node));
    strncpy(result->name, n, 20); // 19 ok too
    result->name[19] = '\0';
    result->assignment = a;
    result->status = "no sub";
    result->next = NULL;  // if missing, must be in both insertions below
    return result;
}

int main() {

    struct node *head;

    // call your function to create a student "Fran Allen" for assignment 1.
    // insert this student at the head of the list

    head = create_student("Fran Allen", 1);
    // head->next = NULL;   not needed, done above


    // not shown here: some number of insertions have been done, no deletions
    // create "liudavid" for A2 and insert at the TAIL of the list

    struct node *tail = head;
    while (tail->next != NULL) {
        tail = tail->next;
    }
    tail->next = create_student("liudavid", 2);
    // tail->next->next = NULL;  // only neeed if missing in create_student

    return 0;
}
```