Question 1. [7 MARKS]

Assume you have a terminal open, and the current working directory contains an executable file called `do_something`.

Part (a) [1 MARK]

Write a shell command that executes the `do_something` program with the argument `now` and with output redirected so that it writes to the file `then` rather than to standard output.

```
./do_something now > then
OR
do_something now > then
```

Part (b) [1 MARK]

Give a one line command to change the permissions on the file `then` so that it maintains its current read and write permissions, but can now be executed by everyone.

```
chmod a+x then     OR    chmod ugo+x then
```

Part (c) [2 MARKS]

Write a one-line shell command that displays the **number** of files in your current directory that contain `CSC209` anywhere in their names. (It would be ok to show some extra information as well.)

```
ls *CSC209* | wc
```

It would be fine to have the -l flag and even the -w flag although that would not work correctly for filesnames with spaces.

Part (d) [1 MARK] In the box, print the number of bytes that will be written to the file by this code.

```
int i[4] = {82, -30, 1000, 4};
fwrite(&i, sizeof(int), 2, fp);
```

sizeof(int) * 2

Part (e) [1 MARK] Give a one-sentence definition of a phony target.

SOLUTION: A Makefile target that does not correspond to an actual file.

Part (f) [1 MARK] Fix the code below so it will compile without error or warnings and works properly.

```
struct stat ss;
int res = stat("filename.txt", &ss);
```

Question 2. [6 MARKS]

For each code fragment below, if the code will not compile or will generate a warning when compiled with the `-Wall` flag, check `COMPILE ERROR` and explain why. If the code will compile, but is not guaranteed to run without an error, check `RUN-TIME ERROR` and explain why. Otherwise, check `NO ERROR` and show what is printed. The first one is done for you.

Code Fragment	ERROR	Output or explanation for error

`int y = 2;`
`int x = y;`
`printf("%d %d", x, y);`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`2 2`

`int x = 5;`
`int *y = &x;`
`x = 12;`
`printf("%d\n%d", x, *y);`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`12 12`

`struct person {`
`  char *name;`
`  int age; };`
`struct person *enigma`
`  = malloc(sizeof(struct person));`
`*(enigma.age) = 42;`
`printf("%d\n", *(enigma.age));`
	☐ NO ERROR
	☑ COMPILE ERROR
	☐ RUN-TIME ERROR
		enigma is a pointer.
		Use notation `(*enigma).age` or `enigma->age`

`char str1[] = "hello";`
`char *str2 = str1;`
`str2[2] = 'y';`
`printf("%s\n", str1);`
`printf("%s", str2);`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		`heylo`
		`heylo`

`char st[20] = "abcd";`
`strncat(st, "ABCDE", 2);`
`printf("%s", st);`
	☑ NO ERROR
	☐ COMPILE ERROR
	☐ RUN-TIME ERROR
		abcdAB

`char *s = "got it";`
`s[0] = 'G';`
`printf("%s", s);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑ RUN-TIME ERROR
		can't assign to string literal

`int **numbers`
`  = malloc(sizeof(int*));`
`int num = 3;`
`numbers[0] = &num;`
`printf("%d",*numbers[0]);`
`free(numbers[0]);`
`free(numbers);`
	☐ NO ERROR
	☐ COMPILE ERROR
	☑ RUN-TIME ERROR
		numbers[0] does not point to heap-allocated memory

Question 3. [7 MARKS]

Consider the string "CSC209". It has 2 substrings of length 5, "CSC20" and "SC209". It has 6 substrings of length 1: "C", "S", "C", "2", "0", "9".

Given an arbitrary string `s` of length `L` and an integer `n`. How many substrings of `s` are of length `n`? Write your answer here: L - n + 1

Complete the function `get_substrings` which given a string `s` and an integer `n`, returns a dynamically-allocated array where the elements of the array are the substrings of `s` of length `n` in the order in which they appear in `s`. If `n` is greater than the length of `s`, `get_substrings` returns `NULL`.

```
char **get_substrings(char *s, int n) {
    if (n > strlen(s))
        return NULL;
    int how_many = strlen(s) - n + 1;

    char **result = malloc(sizeof(char *) * how_many);

    for (int i=0; i < how_many; i++ ) {
        result[i] = malloc(sizeof(char) * (n + 1));
        strncpy(result[i], &s[i], n);
        result[i][n] = '\0';
    }

    return result;
}
```

Question 4. [5 MARKS]

Complete the following program according to the instructions in the comments. Assume that all system calls succeed and that the arguments are of the specified format. Only allocate the space you need. The following code is provided:

```
struct partnership{
  int *share;
  char *p1name;
  char *p2name;
};
```

```
void make_boss(struct partnership *pt) {
    if (pt->share[0] > 50)
        pt->p1name = "boss";
    if (pt->share[1] > 50)
        pt->p2name = "boss";
}
```

```
/* Return the name of the partner with the largest share; If shares are equal, return "equal".*/
char *who_is_boss(struct partnership p) {

    if (p.share[0] > p.share[1])
        return p.p1name;
    if (p.share[0] < p.share[1])
        return p.p2name;
    return "equal";
}

}
/* Set the division of shares for p to 50/50. */
void equalize_division(struct partnership p) {

    p.share[0] = 50;
    p.share[1] = 50;
}

}
int main() {
    struct partnership p;
    p.p1name = "Jack";   p.p2name = "Jill";
    int division[2] = {40, 60};
    p.share = division;

    // Call who_is_boss to return the name of the partner with a larger share
    printf("boss is %s\n", who_is_boss(p));

    // Call the make_boss function (provided above) for p.
    make_boss(&p);

    // Call equalize_division for p.
    equalize_division(p);

}
```