Question 1. [6 MARKS]

Assume you have a terminal open, and the current working directory contains a C file called `imp.c` which includes a header file `imp.h`.

Part (a) [1 MARK]

Write a command to compile `imp.c` into an executable called `imp`, including debugging symbols and using the `gnu99` standard.

```
gcc -g -std=gnu99 -o imp imp.c
```

gcc -g -o imp -std=gnu99 imp.c

Part (b) [1 MARK]

Write a command to run `imp` with `Computer` and `Science` as its command line arguments, also redirecting the program's standard output to a file called `out.txt`.

```
./imp Computer Science > out.txt
OR
imp Computer Science > out.txt
OR
imp Science Computer > out.txt
OR
./imp Science Computer > out.txt
```

./imp Computer Science > out.txt

Part (c) [1 MARK]

*一体的

Write a command to run `imp` with `Computer Science` as a single command line argument (the two words should be separated using a whitespace character).

```
./imp "Computer Science"
OR
imp "Computer Science"
```

./imp "Computer Science"

Part (d) [2 MARKS]

Write a single unix command to display the **number** of unique lines in the file `out.txt` that contain the string `"DEBUG"`.

```
grep DEBUG out.txt | sort | uniq | wc -l
OR
grep DEBUG out.txt | sort -u | wc -l
OR
it is also fine to have either of the above and omit the -l for wc
```

grep "DEBUG" out.txt | sort | uniq | wc -l

Part (e) [1 MARK]

Give a single unix command to set the permissions on the file `imp` so that it is executable by anyone but readable and writeable only by the owner.

```
chmod 711 imp
OR
chmod a+x,u+rw,go-rx imp    // ugly but it works!
```

chmod 711 imp

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.

```
int a[4] = {0, 1, 2, 3};
```

a[0]	a[1]	a[2]	a[3]
0	1	2	3

```
int step = 1;
int *p;

p = a + 1;
*p += step;
```

a[0]	a[1]	a[2]	a[3]
0	2	2	3

```
step++;
p = p + step;
*p = 9;
```

a[0]	a[1]	a[2]	a[3]
0	2	2	9

```
int *q = &a[2];
a[0] = *q + 100;
```

a[0]	a[1]	a[2]	a[3]
102	2	2	9

Question 3. [8 MARKS]

Consider the code and memory diagram below.

Fill in the memory diagram to show the current state of the program exactly before the return statement on **line 12** is executed. For this question, you should assume integers are 4 bytes and pointers are 8. Label the frames on the stack.

```
1  void populate(char *s, int *num, int *p) {
2      int e = strlen(s) - 1;
3      int idx = 0;
4
5      while (e >= 0) {
6          num[idx] = strlen(&s[e]);
7          idx++;
8          e--;
9      }
10
11     *p = 77;
12     return;
13 }
14
15 int main() {
16     char *digits = "12345";
17     int ret;
18     int *arr = malloc(32);
19
20     populate(digits, arr, &ret);
21     // other code not shown
22
23     free(arr);
24     return ret;
25 }
```

Section	Address	Value	Label
Read-only	0x100	1234	
	0x104	5\0	
	0x108		
	0x10c		
	0x110		
	0x114		
	0x118		
	0x11c		
	⋮	⋮	
Heap	0x23c	1	
	0x240	2	
	0x244	3	
	0x248	4	
	0x24c	5	
	0x250	????	
	0x254	????	
	0x258	????	
	0x25c		
	⋮	⋮	
populate	0x458	-1	idx
	0x45c	5	e
	0x460	0x480	p
	0x464		
	0x468	0x23c	nums
	0x46c		
	0x470	0x100	s
	0x474		
main	0x478	0x23c	arr
	0x47c		
	0x480	77	ret
	0x484	0x100	digits
	0x488		

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.

```
struct Coords {
    int x;
    int y;
};
```

```
struct Coords get_coords() {
    struct Coords c;
    c.x = 3;
    c.y = 5;
    return c;
}
```

Part (a) [2 MARKS]

```
struct Coords coords;
coords = get_coords();
printf("(%d, %d)\n", coords.x, coords.y);
```

☑ Works as intended □ Error

(3, 5) (3,5)

% Returning a struct variable results in copying the entire struct byte by byte.

Part (b) [2 MARKS]

```
struct Coords *coords_ptr;
*coords_ptr = get_coords();
printf("(%d, %d)\n", coords_ptr->x, coords_ptr->y);
```

□ Works as intended ☑ Error ☆ Dereference of an initialized pointer

Dereference of an uninitialized pointer.

Question 5. [7 marks]

The question is based on the following linked list definition:

```
struct node {
    int ID;
    char *term; // Points to  a dynamically allocated string.
    struct node *next;
};
```

Implement a function that iterates over the nodes of a linked list starting at the specified `head` and adds the specified `prefix` to the beginning of the term of every node in the list whose `ID` is the `targetID`. For example, if the terms in the list with the required ID were `charge` and `count` and the prefix was `dis`, the new terms would be `discharge` and `discount`. Write your code so that it does not have a memory leak.

```
void add_prefix(struct node *head, char *prefix, int targetID) {
    char *ptr;

    while (head) {
        if (head->ID == targetID) {
         ptr = head->term;

         head->term = malloc(strlen(prefix) + strlen(ptr) + 1);
         strcpy(head->term, prefix);
         strcat(head->term, ptr);

         free(ptr);
        }
        head = head->next;
    }
}
```