Question 1.

Consider the two functions:

```
void set_int1(int x)
{
  x = 42;
}


void set_int2(int *p_x)
{
  *p_x = 42;
}
```

Show how to try to use those functions in `main()`. Explain why one function has an effect but the other one does not.

Consider the following structure.

```
typedef struct student1{
  char name[200];
  char student_number[11];
  int year;
} student1;
```

Question 2.

Write a function that takes in a `student1` and prints their information. Test the function.

Question 3.

Part (a)

Write a function that takes in a pointer to `student1`, and sets `name` to `"Default Student"`

The function signature should be

```
void set_default_name(student1 *p_s)
```

Part (b)

Make sure that a function with a signature like

```
void set_default_name_wrong(student1 s)
```

cannot work.

Now, add setting year to 0 to `set_default_name`. Can this work with `set_default_name_wrong`? Why?

Question 4.

Write a function that uses `malloc` that takes in a pointer to `student *` and sets the pointer to the address of a block of `n_students` students1's.

The function signature should be

```
void create_block1(student1 **p_p_s, int n_students);
```

In your `main` function, show how to use function `create_block1)`.

Question 5.

Write a function with the name `set_name`, which can be used to set the name of a `student1` to a string. If the string is longer than 199 characters, set the name of the input `student1` to the first 199 characters of the input string. Make sure that name is a valid string.

(*Note: input here is used in the sense of data being sent to a function, not data that was received from the input.*)

Use the function `set_name` to set the name of an element from the block you've allocated, and use `printf` to verify that you've successfully used `set_name`

Question 6.

Write the function `destroy_block1` which frees all the memory that was allocated for a block of `student1`s.

Question 7.

Now, consider the structure

```
typedef struct student2{
  char *name;
  char *student_number;
  int year;
} student2;
```

Write `void create_block2(student2 **p_p_s, int num_students)`. Do not allocate the names and student numbers yet. Set both `name` and `student_number` to 0.

Question 8.

Now, write the function `set_name2` which can be used to set the name of a `student2` to an input string. Allocate enough space using `malloc` so that the entire string (including the `NULL`) can be stored.

Use this function with an element of a block allocated using `create_block2`. Verify using `printf` that your function worked correctly.

Question 9.

Write a function named `destroy_block2` which frees all the memory that's been allocated for a block of `student2`s, including any blocks of memory assigned to `names` and `student_numbers`.

Question 10.

Recall from class that if a structure like `student1 s1` is passed into a function, modifying the contents of s1.name would not have an effect outside the function, but if a structure like `student2 s2` is passed into a function, things work differently. Reproduce this behaviour. Now, explain the difference. Show what happens when you pass a pointer to a structure instead of the structure itself.

Question 11.

Use `gcc` to compile and then run a file on ECF. Use

```
gcc -Wall -std=c99 filename.c -o a.exe
```

You should be able to run the executable using

```
./a.exe
```

in the Terminal.

Question 12.

Move the definition of `student1` to a header file, and compile the lab again, after including the header file.

Question 13.

The following code can be used to read in N lines of a text file line-by-line.

```
char line[200];
FILE *fp = fopen(filename, "r");
for(int i = 0; i < N; i++){
  fgets(line, sizeof(line), fp); //read in at most sizeof(line) characters
  //(including '\0') into line.
}
fclose(fp);
```

Write a function that takes in a filename of a text file, and prints the file's contents.

Question 14.

Write a function that takes in a filename that contains integers, one integer per line, and outputs the average of those integers. You can use the library function `atoi`: `https://cplusplus.com/reference/cstdlib/atoi/`.

The function `fgets` returns `NULL` when it has reached the end of the file, so you can use

```
  while(fgets(line, sizeof(line), fp)) {
     ... process line
 }
```

in order to process all the lines in the file.