<u>HW Submission requirements:</u>
1) Put your name and ID number on the top of EACH assignment (in comments for programs)
2) Name each file the name written above each problem
3) Put all the files into a folder and zip it
4) Name the zipped folder with all assignments HW1.zip

MAKE SURE ALL PROGRAMS ARE PROPERLY INDENTED
Your program output should match the given sample run for the program.

Note: the purpose of this HW is to review the C language at the 1310 level. I want to make sure you are comfortable enough with the 1310 C level before we progress to the 1320 C level.

IF YOU HAVE DIFFICULTY WITH THIS HW (SPECIFICALLY THE CONCEPT OF <u>FUNCTIONS, 2D ARRAYS AND FILE IO</u>) RETAKING 1310 MIGHT BE A GOOD IDEA (SINCE THIS COURSE WILL BUILD ON THESE CONCEPTS AND INCREASE IN DIFFICULTY LEVEL).

Problem 1 (10 points) -True/False Submit a document called: Answers.doc

Answer the following true/false questions. You must correctly state **WHY** your answer is true or false in order to receive credit.

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

int main (int argc, char **argv)
{
	int travel_points=1;
	int check=0;

	char answer[20];

	while(!check)
	{
		printf("Enter state, national or international:\n");
		scanf("%s", answer);

		if(!strcmp(answer, "state"))
		{

			travel_points=travel_points+5;
			printf("Adding 5 points. New total: %d\n", travel_points);

		}

		else if (!strcmp(answer, "national"))
		{
```

```
                    travel_points=(travel_points+5)*2;
                    printf("Getting state points plus double points!  New total: %d\n",  travel_points);
            }

            else if(!strcmp(answer, "international"))
            {
                    travel_points*=4;
                    printf("Getting 4x more points!  New total: %d\n", travel_points);
            }

            else if((strcmp(answer, "exit")==0))
            {

                    if(travel_points<100)
                    {
                            printf("Sorry, not enough points for a free trip.\n");

                    }

                    else if(travel_points< 250)
                    {
                            printf("Free dinner at Blue Sushi.\n");
                    }

                    else
                    {
                            printf("Total travel points accumulated: %d.  You get a free trip.\n",
travel_points);
                    }

                    check=1;
            }

            else
            {
                    printf("Not a valid command.\n");

            }

    }

}
```

1. From the code, we can say that the function *strcmp()* has 2 parameters.
2. If the user enters *Exit*, the program will terminate.
3. The only way to stop the *while* loop would be for the user to enter any integer other than 0.
4. Entering the sequence *state, national, international, state*, would result in a free trip for the user.

5. Assuming we don't know where the function declaration of *printf* is, we can deduce from the code that it must be in *stdio.h*.
6. It would be perfectly acceptable to change *char answer[20]* to *char answer[10].*
7. The function *strcmp()* returns an integer.
8. *strcmp(answer, "exit")==0* can be rewritten as *!strcmp(answer, "exit*") without modifying the function of the program.
9. If a user entered the sequence *state* followed *national*, the value of the *total_points* variable would be *21*.
10. This program uses two preprocessor directives and one header.

Problem 2 (45 points)-Write a program. Submit a program called: socialmedia.c

Seven friends have a contest to see who can get the most followers on social media. All users start with a certain number followers (kept in the input file). The program should allow the user to increase the number of followers for the friend that is entered. When the program exits, an output file containing the current number of followers and the winner of the contest should be created (see output file below). The winner gets an amount of five dollars for every follower he has and should be indicated in the output file.

1. Create the following functions:

*/*This function takes a filename of all friends and total number of followers they currently have (see sample input file). The names are read into account_holder and the number of followers are read into followers. The function should return 0 if the file does not open and 1 if all the file info is read in successfully*/*
int get_account_info(char filename[], char account_holder[][20], int followers[]);

*/*This function takes a filename (the file to output to) along with the account names and followers. It should print out to file the same way shown below (outputfile.txt)*/*
void exit_report(char filename[], char account_holder[][20], int followers[])

2. Using the functions you created create a program that matches the sample run and produces the given output file (name of the input and output file given on the command line-see sample run). You can also use the following helper functions if you want (*find_person* and *get_user_input*):

Sample Run:

```
C computer$ ./a.out inputfile.txt outputfile.txt

Add followers or exit?
add
Enter name of person to find:
Levi

How many followers to add?
300
```

Helper functions (you can use these in your program):

```
/*finds a person and returns the index the name was found in…returns -1 if the
    person was not found*/
int find_person(char account_holder[][20], char person[])
{
 int i;

 for(i=0;i<SIZE;i++)
 {
  if (!strcmp(account_holder[i],person))
  {
   return i;
  }
 }
 return -1;
}

/*gets user input */
 void get_user_input(char message[],char answer[])
 {
  printf("%s\n",message);
  fgets(answer,40,stdin);
  strtok(answer,"\n");
 }
```

```
Updated followers for Levi: 700

Add followers or exit?
add
Enter name of person to find:
Levi

How many followers to add?
100
Updated followers for Levi: 800

Add followers or exit?
dunno
unknown response.

Add followers or exit?
add
Enter name of person to find:
David
Person not found.

Add followers or exit?
exit
Outputting file...
```

inputfile.txt

```
Teddy
100
Dave
200
Chauncey
300
Terrell
500
Mark
20
Levi
400
Eric
700
```

Input file structure:
name
initial number of followers

outputfile.txt

```
---Followers for Teddy: 100
---Followers for Dave: 200
---Followers for Chauncey: 300
---Followers for Terrell: 500
---Followers for Mark: 20
---Followers for Levi: 800
---Followers for Eric: 700

--Most followers: Levi with 800 followers!  This user gets $4000.
```

Problem 3 (45 points)-Write a program. Submit a program called: bandpractice.c

You are a band manager trying to get all your members to a specific city to practice. Unfortunately, due to the Covid-19 pandemic, if some members of your band test positive for Covid-19 they will not be able to go to practice. Yesterday, the doctor sent you a list of all the band members and their current status (in a text file called *doctorslist.txt* -see below).

Given the following main, **create the two functions used below to match the sample run and the two output files shown below.** The first should read the doctor's list and save the information. It should return 0 if the file doesn't open and 1 if it does. The program should then output a list of all the positive patients and a list of all the negative patients (in two separate files). One single function should output both files (see main below)- note that it should print to screen all information (see sample run) but should only output to file specifically what the second argument states (*Positive* or *Negative*-see main and sample output files). **DO NOT MODIFY MAIN.**

```
int main (int argc, char** argv) {

  char band_names[SIZE][20];
  char band_diagnose[SIZE];

  int n=read_doctor_list("doctorlist.txt","r",band_names,band_diagnose); /*input file, mode to read file, names of band members, diagnosis as letters (for example, Positive should be saved as just 'P')*/

  if(n)
  {
    band_practice_list("output_p.txt","Positive",band_names,band_diagnose); /*output file, which diagnosis to list (Positive in this case),
names of band members, diagnosis as letters (for example, Positive should be saved as just 'P')*/

    band_practice_list("output_n.txt","Negative",band_names,band_diagnose);
  }
}
```

Sample run:

```
Computers-MacBook-Air:C computer$ ./a.out
Creating list...     /*this is the band_practice_list function*/

Flea: P...Can't go to practice! :(

Anthony: P...Can't go to practice! :(

Chad: N...Good to go to practice! :)

John: N...Good to go to practice! :)

Jack: P...Can't go to practice! :(

Cliff: P...Can't go to practice! :(

Arik: N...Good to go to practice! :)

/*^^^this should print out twice since the same function is called twice*/
```

doctorlist.txt

```
Flea
Positive
Anthony
Positive
Chad
Negative
John
Negative
Jack
Positive
Cliff
Positive
Arik
Negative
```

output_p.txt

```
---List of Positive cases:---
--Flea: +Positive+
--Anthony: +Positive+
--Jack: +Positive+
--Cliff: +Positive+
```

output_n.txt

```
---List of Negative cases:---
-- Chad: -Neg-
-- John: -Neg-
-- Arik: -Neg-
```