CSc 3320: Systems Programming

Fall 2021

Midterm 2: Total points = 200

Assigned: 17th Nov 2021

Submission Deadline: 6th Dec 2021, Monday, 11.59 PM (No extensions. If your submission is not received by this time then it will NOT be accepted.)

Submission instructions:

1. Create a Google doc for your submission.
2. Start your responses from page 2 of the document and copy these instructions on page 1.
3. Fill in your name, campus ID and panther # in the fields provided. If this information is missing TWO POINTS WILL BE DEDUCTED.
4. Keep this page 1 intact. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED.
5. Start your responses to each QUESTION on a new page.
6. If you are being asked to write code, copy the code into a separate txt file and submit that as well. The code should be executable. E.g. if asked for a C script then provide myfile.c so that we can execute that script. In your answer to the specific question, provide the steps on how to execute your file (like a ReadMe).
7. Provide the evidence of your outputs through a screenshot and/or screen video-recordings and copy the same into the document.
8. Upon completion, download a .PDF version of the google doc document and submit the same along with all the supplementary files (videos, pictures, scripts etc).

Full Name: Vivian Do

Campus ID: vdo10

Panther #: 002486640

Dec 06 2021

READ THESE NOTES BEFORE YOU START!

- **Questions 1-4 are 30pts each.**
- **Questions 5 and 6 are 40 pts each.**
- **All questions MUST be ATTEMPTED. Your MIDTERM 2 will NOT be evaluated if there is NO ATTEMPT for even 1 question.**
- **All programs have to be well commented. Non-commented programs will receive 0 points. Comments have to be comprehensible and concise.**

1. Consider the array given below. Write a C program that must be able to sort the elements in the array. You must use pointers in your code to work with the arrays. The sort functionality must be implemented as a separate function named “sort_numeric()”

   Array for your evaluation

   [10, 0.25, -2342, 12123, 3.145435, 6, 6, 5.999, -2, -5, -109.56]

   If given user input A or a: sort in Ascending order

   If given user input D or d: sort in Descending order

File(s): sortNumbers.c
Code:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat sortNumbers.c
// sortNumbers.c
// Midterm 2 Question 1
//
// program to sort the elements of a given array in ascending or descending order

#include <stdio.h>

//main method
int main() {
        double size = 11;
        int i;
        //given array
        double array[] = {10, 0.25, -2342, 12123, 3.145435, 6, 6, 5.999, -2, -5, -109.56};
        char sortingOrder;

        printf("Enter choice to sort array by (A)scending or (D)escending: ");
        scanf(" %c", &sortingOrder);
        sort_numeric(array, size, sortingOrder);
        //printing results
        printf("Array after sorting: \n");
        printArray(array, size);

        return 0;
}

//method to sort array in ascending or descending order
void sort_numeric(double *array, double size, char sortingOrder) {
        int i;
        int j;
```

```
        //checking user input for ascending order
        if (sortingOrder == 'A' || sortingOrder == 'a') {
                for (i = 0; i < size; i++) {
                        for (j = i + 1; j < size; j++) {
                                if (*(array + j) < *(array + i)) {
                                        //calling method to swap elements
                                        swapNumbers((array + i), (array + j));
                                }
                        }
                }
        }
}
        //checking user input for descending order
        else if (sortingOrder == 'D' || sortingOrder == 'd') {
                for (i = 0; i < size; i++) {
                        for (j = i + 1; j < size; j++) {
                                if (*(array + j) > *(array + i)) {
                                        //calling method to swap elements
                                        swapNumbers((array + i), (array + j));
                                }
                        }
                }
        }
}
```

```
//method to swap elements in array
void swapNumbers(double *array, double *swapped) {
        double tempArray;
        tempArray = *(array);
        *(array) = *(swapped);
        *(swapped) = tempArray;
}

//method to print result of array
void printArray(double *array, double size) {
        int i;
        for (i = 0; i < size; i++) {
                printf("%lf\t", array[i]);
        }
        printf("\n");
}
```

*** a separate code file will be submitted as well for every program in this homework!**

Output:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc -o sortNumbers sortNumbers.c
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortNumbers
Enter choice to sort array by (A)scending or (D)escending: a
Array after sorting:
-2342.000000    -109.560000     -5.000000       -2.000000       0.250000        3.145435        5.999000
6.000000        6.000000        10.000000       12123.000000
[vdol0@gsuad.gsu.edu@snowball ~]$
```

```
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortNumbers
Enter choice to sort array by (A)scending or (D)escending: d
Array after sorting:
12123.000000    10.000000       6.000000        6.000000        5.999000        3.145435        0.250000
-2.000000       -5.000000       -109.560000     -2342.000000
[vdol0@gsuad.gsu.edu@snowball ~]$
```

2. Consider the list of names given below. Write a C program that will first create a string array that will contain this list and then sort the elements in the array as per alphabetical order. You must use pointers in your code to work with the arrays. The sort functionality must be implemented as a separate function named “sort_alphabetic()”. The program can be case insensitive (i.e. capital or small letters are treated the same).

   List for your evaluation

   Systems

   Programming

   Deep

   Learning

Internet

Things

Robotics

Course

If given user input A or a: sort in alphabetical order (a comes first) If given user input D or d: sort in reverse alphabetical order(z comes first)

File(s): sortStrings.c

Code:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat sortStrings.c
// sortStrings.c
// Midterm 2 Question 2
//
// program that creates a string array and sorts the elements in alphabetical order, either in ascension or descension order

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

//main method
int main() {
    //given array
    char* array[] = {"Systems", "Programming", "Deep", "Learning", "Internet", "Things", "Robotics", "Course"};
    int size = 8;
    int i;
    char sortingOrder;

    printf("Enter choice to sort array by (A)scending or (D)escending: ");
    scanf(" %c", &sortingOrder);
    sort_alphabetic(array, size, sortingOrder);
    //printing results
    printf("Array after sorting: \n");
    for (i = 0; i < size; i++) {
        printf("%s\n", array[i]);
    }
    return 0;
}
```

```
//method to sort array in ascending or descending order
void sort_alphabetic(char *array[], int size, char sortingOrder) {
    int i;
    int j;
    char* tempArray;
    int compare;

    //checking user input for ascending order
    if (sortingOrder == 'A' || sortingOrder == 'a') {
        for (i = 0; i < size; i++) {
            for (j = i + 1; j < size; j++) {
                compare = strcmp(array[i], array[j]);
                if (compare > 0) {
                    //swapping elements in array
                    tempArray = array[i];
                    array[i] = array[j];
                    array[j] = tempArray;
                }
            }
        }
    }
```

```
    //checking user input for descending order
    else if (sortingOrder == 'D' || sortingOrder == 'd') {
        for (i = 0; i < size; i++) {
            for (j = i + 1; j < size; j++) {
                compare = strcmp(array[i], array[j]);
                if (compare < 0) {
                    //swapping elements in array
                    tempArray = array[i];
                    array[i] = array[j];
                    array[j] = tempArray;
                }
            }
        }
    }
}
```

Output:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc -o sortStrings sortStrings.c
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortStrings
Enter choice to sort array by (A)scending or (D)escending: a
Array after sorting:
Course
Deep
Internet
Learning
Programming
Robotics
Systems
Things
[vdol0@gsuad.gsu.edu@snowball ~]$
```

```
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortStrings
Enter choice to sort array by (A)scending or (D)escending: d
Array after sorting:
Things
Systems
Robotics
Programming
Learning
Internet
Deep
Course
[vdol0@gsuad.gsu.edu@snowball ~]$
```

3. Repeat Question 1 or Question 2, considering that the number of elements can potentially increase. That is, the size of the array will be unknown at the start of the program. Note that the requirement of using pointers still holds.

Show proof of evaluation of your program being able to work for more than 10 entries. Show 5 evaluation trials in your submission. You can pick any number of entries between 10 and 30 for your trials.

(Hint: *To solve this, use dynamic memory allocation, where you will NOT treat the input array as a known or finite size. Allocate memory space (e.g. malloc()) as and when the number of elements in the list increases).*

File(s): sortStringsDynamic.c
Code:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat sortStringsDynamic.c
// sortStringsDynamic.c
// Midterm 2 Question 3
//
// repeating question 2, but the size of the array is unknown at the start instead of being given an array

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

//main method
int main() {
    //using dynamic memory allocation to create array
    char **array = malloc(1);
    int size = 0;
    char input[15];
    int i;
    char sortingOrder;

    //prompting user to input words into array or q to stop
    printf("Enter words or (Q)uit: \n");
    do {
        scanf("%s", input);
        array = (char **)realloc(array, (size + 1) * sizeof(char *));
        array[size++] = strdup(input);
    }
    while (strcasecmp(input, "q") != 0);
```

```
    printf("Enter choice to sort array by (A)scending or (D)escending: ");
    scanf(" %c", &sortingOrder);
    sort_alphabetic(array, size - 1, sortingOrder);
    //printing results
    printf("Array after sorting: \n");
    for (i = 0; i < size; i++) {
        printf("%s\n", array[i]);
    }

    //deallocating memory for every element in array
    for (i = 0; i < size; i++) {
        free(array[i]);
    }
    //deallocating array
    free(array);
    return 0;
}

//method to sort array in ascending or descending order
void sort_alphabetic(char *array[], int size, char sortingOrder) {
    int i;
    int j;
    char* tempArray;
    int compare;
```

```
    //checking user input for ascending order
    if (sortingOrder == 'A' || sortingOrder == 'a') {
        for (i = 0; i < size; i++) {
            for (j = i + 1; j < size; j++) {
                compare = strcmp(array[i], array[j]);
                if (compare > 0) {
                    //swapping elements in array
                    tempArray = array[i];
                    array[i] = array[j];
                    array[j] = tempArray;
                }
            }
        }
    }

    //checking user input for descending order
    else if (sortingOrder == 'D' || sortingOrder == 'd') {
        for (i = 0; i < size; i++) {
            for (j = i + 1; j < size; j++) {
                compare = strcmp(array[i], array[j]);
                if (compare < 0) {
                    //swapping elements in array
                    tempArray = array[i];
                    array[i] = array[j];
                    array[j] = tempArray;
                }
            }
        }
    }
}

[vdol0@gsuad.gsu.edu@snowball ~]$
```

Output:

Trial 1: Airplanes Song Lyrics (13 words & Ascending)

```
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc -o sortStringsDynamic sortStringsDynamic.c
sortStringsDynamic.c:35:6: warning: conflicting types for 'sort_alphabetic' [enabled by default]
 void sort_alphabetic(char *array[], int size, char sortingOrder) {
      ^
sortStringsDynamic.c:22:5: note: previous implicit declaration of 'sort_alphabetic' was here
     sort_alphabetic(array, size - 1, sortingOrder);
     ^
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortStringsDynamic
Enter words or (Q)uit:
can
we
pretend
that
airplanes
in
the
night
sky
are
like
shootin'
stars
q
```

```
Array after sorting:
airplanes
are
can
in
like
night
pretend
shootin'
sky
stars
that
the
we
q
[vdol0@gsuad.gsu.edu@snowball ~]$
```

Trial 2: Rubia Song Lyrics (21 words & Descending)

```
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortStringsDynamic
Enter words or (Q)uit:
life
blooms
like
a
flower
far
away
or
by
the
road
waiting
for
the
one
to
find
the
way
back
home
q
Enter choice to sort array by (A)scending or (D)escending: d
```

```
Array after sorting:
way
waiting
to
the
the
the
road
or
one
like
life
home
for
flower
find
far
by
blooms
back
away
a
q
[vdol0@gsuad.gsu.edu@snowball ~]$
```

Trial 3: Nier:Automata Ending A Quote (16 words & Ascending)

```
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortStringsDynamic
Enter words or (Q)uit:
what
is
it
that
separates
machines
from
androids
like
us?
the
machines
has
gained
emotions...
consciousness.
q
Enter choice to sort array by (A)scending or (D)escending: a
```

```
Array after sorting:
androids
consciousness.
emotions...
from
gained
has
is
it
like
machines
machines
separates
that
the
us?
what
q
[vdol0@gsuad.gsu.edu@snowball ~]$
```

Trial 4: Nier:Automata Ending B Quote (27 words & Descending)

```
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortStringsDynamic
Enter words or (Q)uit:
and
so,
the
final
battle
with
adam
and
eve
came
to
an
end.
this
battle
will
likely
have
a
geat
effect
on
the
outcome
of
the
war.
q
Enter choice to sort array by (A)scending or (D)escending: d
```

```
Array after sorting:
with
will
war.
to
this
the
the
the
so,
outcome
on
of
likely
have
geat
final
eve
end.
effect
came
battle
battle
and
and
an
adam
a
q
[vdol0@gsuad.gsu.edu@snowball ~]$
```

Trial 5: Random Word Generator (15 words & Ascending)

```
[vdol0@gsuad.gsu.edu@snowball ~]$ ./sortStringsDynamic
Enter words or (Q)uit:
nourish
obsequious
bike
polite
tangible
gruesome
inform
ruthless
tee
pocket
hobbies
money
redundant
prison
simple
q
Enter choice to sort array by (A)scending or (D)escending: a
```

```
Array after sorting:
bike
gruesome
hobbies
inform
money
nourish
obsequious
pocket
polite
prison
redundant
ruthless
simple
tangible
tee
q
[vdol0@gsuad.gsu.edu@snowball ~]$
```

4. Using C programming and using Structures or Unions in your program, build a COVID vaccine registration form where any user can register by filling in their First Name, Last Name, Date of Birth (mm/dd/yyyy), Dose number (1 or 2), Date of previous dose, Type of vaccine (e.g. Pfizer, Moderna, Johnson &

Johnson etc.), Residential zip code.

Upon registration, the system must output a 8 letter alphanumeric code that will be unique to that user. The code is generated as <First letter of First Name><First Letter of Last Name><current age of user -as of registration date><First letter of Vaccine type><last 3 numbers of zipcode>

Add functionality in your program such that it will display all the user's information on the screen (one item in each line).

Show an evaluation trial for registering at least 10 users. For registration, ,for relevant questions, users must choose values based on the options provided. Use pseudo values instead of actual personal details.
(Hint: Write a program that contains main(), register(), generate_code() and retrieve() functions, at the least).

File(s): covidForm.c
Code:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat covidForm.c
// covidForm.c
// Midterm 2 Question 4
//
// program building a covid vaccine registration form
// upon registration, system will output a unique 8-letter alphanumeric code

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

//constructing user
struct User {
        char firstName[15];
        char lastName[15];
        char bday[10];
        char age[3];
        char gender[10];
        char doseNum[2];
        char previousDose[10];
        char vaccineType[15];
        char zipCode[5];
        char userID[9];
};

void registered(struct User);
void generate_code(struct User);
void retrive(struct User);
void displayForm();
```

```
//main method
int main() {
	//creating variables
	int userInput;
	struct User users[20];
	int numberOfUsers = 0;
	char *idInput;
	int i;

	//do-while loop prompting user to select option for registration form
	do {
		//displaying form
		displayForm();
		scanf("%d", &userInput);

		//switch-case for selecting options
		switch(userInput) {
			case 1:
				if (numberOfUsers >= 20) {
					printf("List is full.");
					break;
				}
			registered(users[numberOfUsers]);
			numberOfUsers++;
			break;
			case 2:
				printf("User ID: ");
				scanf("%s", idInput);
				for (i = 0; i < numberOfUsers; i++) {
					if (strcasecmp(idInput, users[i].userID) == 0) {
						retrive(users[i]);
					}
```

```
					else {
						printf("The ID you've entered does not exist.\n");
					}
				}
				break;
			case 3:
				break;
			default:
				printf("ERROR! Could not read input.\n");
				break;
		}
	}
	while (userInput != 3);
	return 0;
}
```

```
//method to display contents of the form
void displayForm() {
        printf("\n - COVID Vaccine Registration Form - \n\n");
        printf("\t1\tRegister a User\n");
        printf("\t2\tDisplay Info\n");
        printf("\t3\tQuit\n\n");
        printf("\tEnter a Number: ");
}

//method to save user information in struct based on user input
void registered(struct User users) {
        char input[15];
        printf("First Name: ");
        scanf("%s", &input);
        strcpy(users.firstName, input);

        printf("Last Name: ");
        scanf("%s", &input);
        strcpy(users.lastName, input);

        printf("Date of Birth (mm/dd/yyyy): ");
        scanf("%s", &input);
        strcpy(users.bday, input);
```

```
        printf("Current Age: ");
        scanf("%s", input);
        strcpy(users.age, input);

        printf("Gender: ");
        scanf("%s", input);
        strcpy(users.gender, input);

        printf("Dose Number (1 or 2): ");
        scanf("%s", input);
        strcpy(users.doseNum, input);

        printf("Date of Previous Dose (mm/dd/yyy) or N/A: ");
        scanf("%s", input);
        strcpy(users.previousDose, input);

        printf("Type of Vaccine (Pfizer, Moderna, or Johnson&Johnson): ");
        scanf("%s", input);
        strcpy(users.vaccineType, input);
```

```
        printf("Zip Code: ");
        scanf("%s", input);
        strcpy(users.zipCode, input);

        //generates the unique alphanumeric code for user
        generate_code(users);
}

//code to generate user code
//based on first initial, last initial, current age, vaccine type initial, and last 3 numbers of zipcode
void generate_code(struct User users) {
        users.userID[0] = users.firstName[0];
        users.userID[1] = users.lastName[0];
        users.userID[2] = users.age[0];
        users.userID[3] = users.age[1];
        users.userID[4] = users.vaccineType[0];
        users.userID[5] = users.zipCode[2];
        users.userID[6] = users.zipCode[3];
        users.userID[7] = users.zipCode[4];
        users.userID[8] = '\0';

        printf("User ID: %s\n", users.userID);
}
```

```
//method to retrieve user inputted information
void retrive(struct User users) {
        printf("\nFirst Name: %s", users.firstName);
        printf("\nLast Name: %s", users.lastName);
        printf("\nDate of Birth: %s", users.bday);
        printf("\nCurrent Age: %s", users.age);
        printf("\nGender: %s", users.gender);
        printf("\nDose Number: %s", users.doseNum);
        printf("\nDate of Previous Dose: %s", users.previousDose);
        printf("\nType of Vaccine: %s", users.vaccineType);
        printf("\nZip Code: %s\n", users.zipCode);
}

[vdol0@gsuad.gsu.edu@snowball ~]$
```

Output:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc -o covidForm covidForm.c
[vdol0@gsuad.gsu.edu@snowball ~]$ ./covidForm

 - COVID Vaccine Registration Form -

        1       Register a User
        2       Display Info
        3       Quit

        Enter a Number: 1
First Name: Vivian
Last Name: Do
Date of Birth (mm/dd/yyyy): 02/11/2001
Current Age: 20
Gender: female
Dose Number (1 or 2): 1
Date of Previous Dose (mm/dd/yyy) or N/A: n/a
Type of Vaccine (Pfizer, Moderna, or Johnson&Johnson): Moderna
Zip Code: 30043
User ID: VD20M043

 - COVID Vaccine Registration Form -

        1       Register a User
        2       Display Info
        3       Quit

        Enter a Number:
```

```
        Enter a Number: 1
First Name: Stevie
Last Name: Vu
Date of Birth (mm/dd/yyyy): 04/18/2001
Current Age: 20
Gender: male
Dose Number (1 or 2): 2
Date of Previous Dose (mm/dd/yyy) or N/A: 06/20/2021
Type of Vaccine (Pfizer, Moderna, or Johnson&Johnson): Pfizer
Zip Code: 30341
User ID: SV20P341

 - COVID Vaccine Registration Form -

        1       Register a User
        2       Display Info
        3       Quit

        Enter a Number:
```

```
        Enter a Number: 2
User ID: hwllJl23
The ID you've entered does not exist.
The ID you've entered does not exist.

 - COVID Vaccine Registration Form -

        1       Register a User
        2       Display Info
        3       Quit

        Enter a Number: 3
[vdol0@gsuad.gsu.edu@snowball ~]$
```

5. Copy the contents of this document into a text file. Make sure the spacings and indentations are included. Write a C program that READS the TEXT file and then outputs

    a. the number of characters (space is to be considered a character),

    b. number of words (a word is any sequence of non-white-space characters)

    c. number of lines.

-- Each of the functionalities a, b, and c above must be written as FUNCTIONS and passing of arguments MUST be through POINTERS.

-- Name the functions problem5char.c, problem5words.c, and problem5lines.c

-- Write a Makefile that will execute the main C program to include all these three scripts.

-- All these outputs from (number of chars, words and lines) must be saved into ANOTHER text file row-wise. Every execution of your script with a new input must APPEND the outputs to a new row in that text file. You can separate each value in a row using any delimiter of your choice (e.g. comma or semi-colon etc)

File(s): midterm2.txt, question5.c, problem5char.c, problem5words.c problem5lines.c, makefile

Code:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat question5.c
// question5.c
// Midterm 2 Question 5
//
// program that counts the letters, words, and lines of the Midterm document
// this is the main execution program that prints the results of the functions

#include <stdio.h>

int main() {
        //directing filepath to the midterm document
        char *filepath = "midterm2.txt";

        //getting letter, word, and line count from functions
        int charas = countCharacters(filepath);
        int words = countWords(filepath);
        int lines = countLines(filepath);

        //printing results
        printf("Characters: %d\nWords: %d\nLines: %d\n", charas, words, lines);
}
[vdol0@gsuad.gsu.edu@snowball ~]$
```

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat problem5char.c
//function to count the number of letters in Midterm document

#include <stdio.h>

//method to count letters
int countCharacters(char* filepath) {
        //opening & reading file
        FILE* file = fopen(filepath, "r");
        //variable to count letters
        int count = 0;
        char chara;

        //going through entire file until eof
        while ((chara = fgetc(file)) != EOF) {
                //checking if letter is not a newline chara
                if (chara != '\n') {
                        count++;
                }
        }
        //closing file
        fclose(file);
        return count;
}
[vdol0@gsuad.gsu.edu@snowball ~]$
```

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat problem5words.c
//function to count the number of words in Midterm document

#include <stdio.h>

//method to count words
int countWords(char* filepath) {
        //opening & reading file
        FILE* file = fopen(filepath, "r");
        //variable to count letters
        int count = 0;
        char word;

        //going through entire file until eof
        while ((word = fgetc(file)) != EOF) {
                //checking is word is a newline or space chara
                if (word == '\n' || word == ' ') {
                        count++;
                }
        }
        //closing file
        fclose(file);
        return count;
}
[vdol0@gsuad.gsu.edu@snowball ~]$
```

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat problem5lines.c
//function to count the number of lines in Midterm document

#include <stdio.h>

//method to count lines
int countLines(char* filepath) {
        //opening & reading file
        FILE* file = fopen(filepath, "r");
        //variable to count
        int count = 1;
        char line;

        //going through entire file until eof
        while ((line = fgetc(file)) != EOF) {
                //checking if line is a newline chara
                if (line == '\n') {
                        count++;
                }
        }
        //closing file
        fclose(file);
        return count;
}
[vdol0@gsuad.gsu.edu@snowball ~]$
```

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat makefile
#makefile that executes the main program that includes all the functions

question5: question5.o problem5char.o problem5words.o problem5lines.o
        gcc -o question5 question5.o problem5char.o problem5words.o problem5lines.o

question5.o: question5.o problem5char.o problem5words.o problem5lines.o
        gcc -c question5.c problem5char.c problem5words.c problem5lines.c

problem5char.o: problem5char.c
        gcc -c problem5char.c

problem5words.o: problem5words.c
        gcc -c problem5words.c

problem5lines.o: problem5lines.c
        gcc -c problem5lines.c

[vdol0@gsuad.gsu.edu@snowball ~]$
```

Output:

first compiling it by gcc -o

```
[vdol0@gsuad.gsu.edu@snowball ~]$ gcc -o question5 question5.c problem5char.c problem5words.c problem5lines.c
[vdol0@gsuad.gsu.edu@snowball ~]$ ./question5
Characters: 5676
Words: 1046
Lines: 74
[vdol0@gsuad.gsu.edu@snowball ~]$
```

then compiling by makefile

```
[vdol0@gsuad.gsu.edu@snowball ~]$ make
make: Circular question5.o <- question5.o dependency dropped.
gcc -c question5.c problem5char.c problem5words.c problem5lines.c
gcc -o question5 question5.o problem5char.o problem5words.o problem5lines.o
[vdol0@gsuad.gsu.edu@snowball ~]$ ./question5
Characters: 5676
Words: 1046
Lines: 74
[vdol0@gsuad.gsu.edu@snowball ~]$
```

6. Repeat everything in Problem 5, but create the functionalities a, b and c as HEADER (Library) files and execute using a Make file.

File(s): problem5char.h, problem5words.h, problem5lines.h, and everything from question 5
Code:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat problem5words.h
//library header for counting words
#ifndef PROBLEM5WORDS_H_INCLUDED
#define PROBLEM5WORDS_H_INCLUDED

int problem5words(FILE *filepath) {
        int count = 0;
        char word;

        while ((word = fgetc(filepath)) != EOF) {
                if (word == '\n' || word == ' ') {
                        count++;
                }
        }
        return count;
}

#endif
```


```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat problem5words.h
//library header for counting words
#ifndef PROBLEM5WORDS_H_INCLUDED
#define PROBLEM5WORDS_H_INCLUDED

int problem5words(FILE *filepath) {
        int count = 0;
        char word;

        while ((word = fgetc(filepath)) != EOF) {
                if (word == '\n' || word == ' ') {
                        count++;
                }
        }
        return count;
}

#endif
```


```
[vdol0@gsuad.gsu.edu@snowball ~]$ cat problem5lines.h
//library header for counting lines
#ifndef PROBLEM5LINES_H_INCLUDED
#define PROBLEM5LINES_H_INCLUDED

int problem5lines(FILE *filepath) {
        int count = 1;
        char line;

        while ((line = fgetc(filepath)) != EOF) {
                if (line == '\n') {
                        count++;
                }
        }
        return count;
}

#endif
```

Output:

```
[vdol0@gsuad.gsu.edu@snowball ~]$ vi problem5char.h
[vdol0@gsuad.gsu.edu@snowball ~]$ vi problem5words.h
[vdol0@gsuad.gsu.edu@snowball ~]$ vi problem5lines.h
[vdol0@gsuad.gsu.edu@snowball ~]$ make
make: Circular question5.o <- question5.o dependency dropped.
gcc -c question5.c problem5char.c problem5words.c problem5lines.c
gcc -o question5 question5.o problem5char.o problem5words.o problem5lines.o
[vdol0@gsuad.gsu.edu@snowball ~]$ ./question5
Characters: 5676
Words: 1046
Lines: 74
[vdol0@gsuad.gsu.edu@snowball ~]$
```