SFWRENG 2XC3- Major lab 4 - Lab section L03

Rules for Major Labs:

1. For major labs, students must be taking part in the registered lab section and should not move between other lab sections.

2. All major labs are an open book; use of all resources printed, written, or from anywhere on the Internet is permitted, but using ChatGPT and any AI-based language models is not allowed.

3. You are not allowed to solicit help from other persons or use work of other persons for even a part of the solution or discuss the solution with other persons. Especially, you are not allowed to discuss the assignment problems with the fellow students. In simple terms, the entire solution must be your own work.

4. You can submit your work (even partial work) as many times as you wish at any time from the posting of the assignment to the assignment's deadline; all submissions are saved in the repository. For marking, the latest submission is used.

5. An integral part of the submission process of your work is for you to check the submission.

6. Files saved in your Avenue account (Dropbox) are considered submitted.

7. After the assignment's deadline, during lab session time, no submission via the email or via Avenue is possible.

8. If submission is not possible for whatever reason, you could submit it to the TA and CC me, my email address: yazdinea@mcmaster.ca

After the assignment's deadline, during lab session time, no submission via the email or via Avenue is possible.

In this lab, there are two tasks and two deliverables: text files (C programs) **mprog3.c** for the first task, and **Prog_Tri.c** for the second task. Creating a screenshot of the output of your codes is recommended in a Word or PDF file.

Task 1. C program named mprog3.c [15 marks, Main 5 marks and each function has 2 marks**]**

Description of the C program mprog3.c

1. The program has to have the following functions:

    - void init_arr(int x[],int len)
    - void rot_left_arr(int x[],int len)
    - void rot_right_arr(int x[],int len)
    - void rot_arr(int x[],int len,int r)
    - void show_arr(int x[],int len)
    - int main(int argc,char** argv)

    You need to know that

    - init_arr(int x[],int len) is used to randomly initialize an integer array x , **this function's code is given here. so, you do not program it, you just use it in your program while add the library time, #include <time.h> in top your code. However you need to place a prototype declaration** void init_arr(int x[], int len); **somewhere at the top of your program (before all the other functions is the best place).**

```
void init_arr(int x[],  int len) {

srand(time(NULL)); // Seed the random number generator with the current time

int i;

for (i = 0; i < len; i++) {

x[i] = rand() % 100; // Generate random numbers between 0 and 99

} }
```

    - rot_left_arr(int x[], int len): This function is used to rotate the array x one position to the left. In this operation, the first element of the array is moved to the end, and all other elements are shifted one position to the left. For example, if the array is [1, 2, 3, 4], after applying rot_left_arr, it would become [2, 3, 4, 1].
    - rot_right_arr(int x[], int len): This function is used to rotate the array x one position to the right. Here, the last element of the array is moved to be the first, and all other elements are shifted one position to the right. For instance, if the array is [1, 2, 3, 4], after applying rot_right_arr, it would turn into [4, 1, 2, 3].
    - rot_arr(int x[],int len,int r) is used to rotate the array r positions to the right (if r > 0), or -r positions to the left (if r < 0). This function just iterates rot_left_arr() or rot_right_arr() the required number of times.
    - show_arr(int x[],int len) displays the array x on the screen.
2. The function main() has a (static) integer array of size 20 , let us call it x , but you can call it whatever you want.

3. The program checks the number of command line arguments. If it is not 3, it displays an error message and terminates (see the sample runs below).
4. The first command line argument, argv[1], is the length of the array (since it is a string, it must be converted to the number, most conveniently by atoi() standard function). If the length < 2 or > 20, the program displays an error message and terminates (see below a sample run), let us store the number in an integer variable len , but you can call it whatever you want. If it is OK, it displays a message about the rotation (see the sample runs below).
5. The second command line argument argv[2] is the number of positions of the rotation (again, since it is a string, it must be converted to a number), let us store the number in an integer variable r , but you can call it whatever you want.
6. Then it calls the function init_arr(x,len) .
7. Then the program displays the array by a call to show_arr(x,len)
8. Then it calls the function rot_arr() and passes it x , len , and r
9. Then the program displays the modified array by a call to show_arr(x,len)
10. The program is compiled by gcc -o mprog3 mprog3.c

Sample Run:

```
[yazdinea@moore ~/M6] nano mprog3.c
[yazdinea@moore ~/M6] gcc -o mprog3 mprog3.c
[yazdinea@moore ~/M6] ./mprog3
incorrect number of command line arguments
usage: mprog3 <size> <rotation>
[yazdinea@moore ~/M6] ./mprog3 30 -5
incorrect length 30
[yazdinea@moore ~/M6] ./mprog3 6 -3
creating array of size 6, rotating it left by 3 positions
2 98 88 98 76 74
98 76 74 2 98 88
[yazdinea@moore ~/M6] ./mprog3 7 2
creating array of size 7, rotating it right by 2 positions
92 96 76 50 9 78 27
78 27 92 96 76 50 9
[yazdinea@moore ~/M6]
```

Task2: Recursive Tribonacci Sequence Calculation, Prog_Tri.c [4 Marks]

Objective: Write a C program, **Prog_Tri.c,** that calculates the nth term of the Tribonacci sequence using a recursive approach. The integer n will be provided by the user.

Task Details:

1. **Function Development:**
    - **Write a function named tribonacci that calculates the nth term of the Tribonacci sequence.**
    - **Function Signature: int tribonacci(int n);**
    - **The function should be recursive. The Tribonacci sequence is defined as:**

- $T(0) = 0$
- $T(1) = 1$
- $T(2) = 1$
- $T(n) = T(n-1) + T(n-2) + T(n-3)$ for $n > 2$

2. **Main Function:**
    - In the **main** function, prompt the user with the message: **Please enter a non-negative integer to find the nth term of the Tribonacci sequence**:
    - Capture the user's input into an integer variable, say **term**.
    - Call the **tribonacci** function to compute the nth term and display the result to the user.
3. **Compilation & Execution:**
    - Your program should compile without any errors.
    - When executed, the program should prompt the user for input, then calculate and display the nth term of the Tribonacci sequence.

Hints:

- **Recursive definition of Tribonacci:**
    - $T(0) = 0$, $T(1) = 1$, $T(2) = 1$
    - $T(n) = T(n-1) + T(n-2) + T(n-3)$ for $n > 2$
- Ensure you handle base cases for **n = 0, n = 1, and n = 2** in your recursive function.

Sample Run:

To find $T(8)$:

$T(8) = T(7) + T(6) + T(5)$

$T(8) = 24 + 13 + 7 = 44$

```
[yazdinea@moore ~/M4] ./Prog_Tri
Please enter a non-negative integer to find the nth term of the Tribonacci sequence: 8
The 8th term of the Tribonacci sequence is: 44
[yazdinea@moore ~/M4]
```