Computer Science and Engineering Department
Indian Institute of Technology Kharagpur

Programming and Data Structure Laboratory: CS19003
1st Year: 1st Semester

Assignment – 2: Conditional Branching, and Loops *Marks: 100*
Assign Date: *August 20, 2024* Submit Date: *16:45, August 20, 2024*

1 General Instruction

- Create a folder with folder name *RollNo_Asgn*2.
- Give the name of the program as $< p >$ .c where $< p >$ implies the problem number, like 1.c, 2.c, 3.c, etc. Store all the programs of this week under this directory.
- Zip the folder *RollNoAsgn*1 and upload the zip file to the Moodle course web page latest by 4:45 PM (without penalty). The cutoff time will be till 5:00 PM with a penalty of 25% on your secured marks (i.e., if you secured 80 marks, after penalty you will get 60 marks). Beyond 5:00 PM, the moodle system will not allow you to submit, as a result you will get zero.
- Attemp all the problems. All the problems are of equal marks.
- There will be ZERO tolerance for plagiarism (copy from anywhere) cases.

2 Credits

- Problem 1: **30**
  - Input/Output: **10** [3+7]
  - Loop for equation 1: **10**
  - Loop for equation 2: **10**
- Problem 2: **30**
  - Input/Output: **10** [3+7]
  - Outer Loop: **10**
  - Nested Loop: **10**
- Problem 3: **40**
  - Input/Output (conversion to BLOCK, and skipping whenever is required): **20** [5+15]
  - Switch-case: **10**
  - do-while: **10**

3 Problem 1

The life of π was never easy. Despite the fact that it is defined as $\frac{22}{7}$, it never converges. Every time, we divide, we generate a new digit at the next decimal place, where there is no pattern. Nevertheless, there are several ways to compute the value of π, apart from division of 22 by 7.

For example,

$$\frac{\pi^2}{6} = \sum_{i=1}^{\infty} \frac{1}{i^2} \tag{1}$$

$$\pi = \sum_{i=1}^{\infty} \frac{(-1)^{(i+1)} \times 4}{2 \times i - 1} \tag{2}$$

Write down a C program (use *while* loop) to compute the value of π using above two definitions. Do not use any math library function, or any other user defined function in your program. Report the number of iterations required for each of the equations so that the result is correct upto n decimal places. Read user input n, where $9 > n > 0$. Use required format specifier to print upto n decimal places. Consider that $\frac{22}{7}$ is the correct answer to estimate your error.

NOTE: In order to output a correct result upto 4 decimal places, there should not be any change upto 4 decimal places between two consecutive executions.

4 Problem 2

The natural logarithm base e can also be computed as:

$$\frac{1}{e} = \sum_{i=0}^{\infty} \frac{(-1)^i}{i!} \tag{3}$$

Write down a C program (use for loop, if required do nesting of the loops) which does not use any math library function, or any other user defined function to compute the value of e using the above equation. Report your result correct upto n decimal places (no change in value upto n decimal places between two consecutive iterations). Use required format specifier to print upto n decimal places. Read n from user input. Make sure that $8 > n > 1$. In case of any wrong user input (not correct as per the mentioned specification), iterate for the correct input (until you get any).

5 Problem 3

One of the basic operarations in a scanning process is to categorize the character it scans. A hypothetical scanner follows a set of rules as mentioned below:

Category	**Character**
Vowel	a e i o u
Consonant	Other English alphabet except a e i o u
Digit	0 1 2 3 4 5 6 7 8 9
BinaryOperator	+ - * / %
Punctuator	, : ; ‘ ‘ “ ”
Bracket	( ) { } [ ]
No Action (no need to write anything)	Not covered in the above rows

Write a C program using $switch - case$ and $do - while$ loop that will read characters from the standard input, and will print the character along with in which category it belongs in a new line.

In case of English alphabet, it will convert and print BLOCK letter. The program will keep on continuing until somebody hits enter button. Upon reading new line character, the program will terminate normally.

NOTE:

- ASCII code for ‘A’ in decimal is 65
- ASCII code for ‘a’ in decimal is 97
- ASCII code for ‘0’ in decimal is 48

The next character or the digit is the next ASCII code.

Sample Input/Output:

- Input: *(a+c)*E*

  ```
  $./a.out
  (a+c)*E
  Bracket (
  Vowel A
  BinaryOpreator +
  Consonant C
  Bracket )
  BinaryOperator *
  Vowel E
  $
  ```

- *Input: Myself, ”roHAN”, a Kgpian.*

  ```
  $./a.out
  Myself, "roHAN", a Kgpian.
  Consonant M
  Consonant Y
  Consonant S
  Vowel E
  Consonant L
  Consonant F
  Punctuator ,
  Punctuator "
  Consonant R
  Vowel O
  Consonant H
  Vowel A
  Consonant N
  Punctuator "
  Punctuator ,
  Vowel A
  Consonant K
  Consonant G
  Consonant P
  Vowel I
  Vowel A
  Consonant N
  $
  ```