Section 14

PDS Lab **Lab - 11** **25.01.2023**

Instructions:

Give sufficient comment against each statement in your program.
You should save each program with the file name as specified against each problem.
There is a partial credit even if your program does not run successfully for all the test cases as mentioned.

Q1. Write a **recursive** function to print the sum of harmonic series up to n.
(Only those programs that **use recursion to generate the sum** will be awarded marks)
(assume $0 \leq n \leq 99999$)
Harmonic series up to n

$$Sum(H_n) = \left(\frac{1.0}{1} + \frac{1.0}{2} + \frac{1.0}{3} + \cdots + \frac{1.0}{n}\right)$$

or,

$$Sum(H_n) = \sum_{i=1}^{n} \frac{1.0}{i}$$

[Note:
Division in C is bugged,
if you divide 1/*n* where *n* is an int,
then we get an integer output which is usually 0 for n>1,
as a work around use
(1.0/n) or
(1/(float)n)
every time)
]

Test cases:

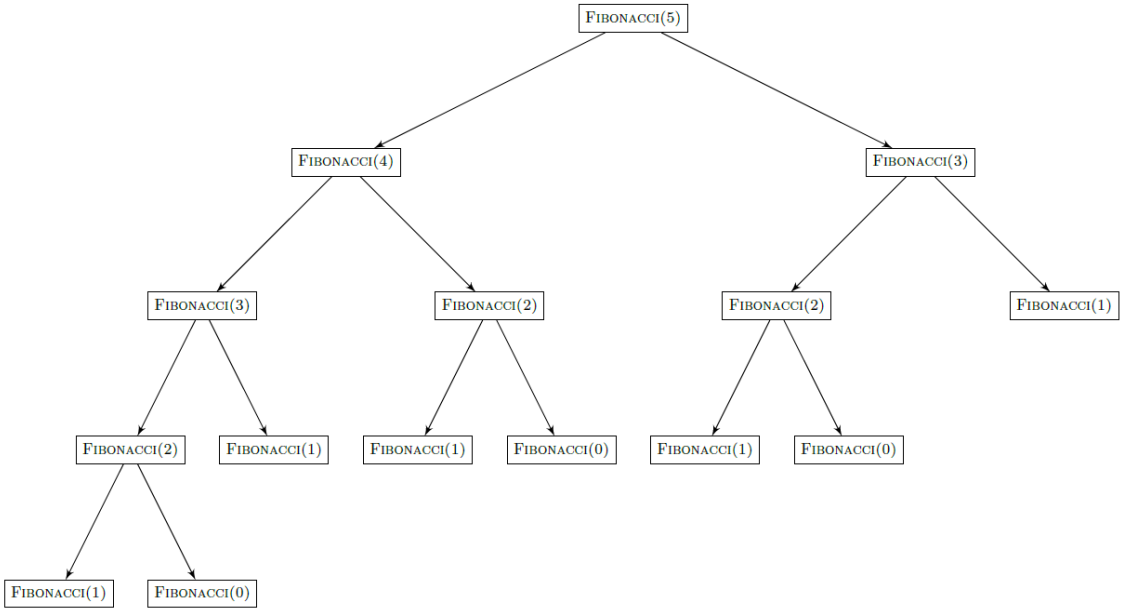
#	INPUT	OUTPUT
1	1	1.000
2	5	2.283
3	8	2.718
4	10	2.929

[Time: 20 minutes]

Q2. Write a **recursive** function to check if a given number n, belongs to the Fibonacci sequence.

(Only those programs that **use recursion to generate the Fibonacci seq** will be awarded marks)
[Assume $[0 \leq n \leq 100]$]
[Hint: Recursion will follow this example]
The Fibonacci sequence can be defined by the following three equations:

$$F_0 = 0 \quad \text{(applies only to the first integer)}$$
$$F_1 = 1 \quad \text{(applies only to the second integer)}$$
$$F_n = F_{n-1} + F_{n-2} \quad \text{(applies to all other integers)}$$



Test cases:

#	INPUT	OUTPUT
1	3	Yes
2	10	No
3	13	Yes
4	0	Yes

[Time: 20 minutes]

Q3. Write a program that will insert a new value to an already existing **sorted linked list** such that the ordering is maintained.

You can assume the following:

a) The linked list is to be maintained in ascending order, for example, 1, 2, 3, …
b) An input is given where all elements are already in sorted order.

User Input: Number of elements in the array, say *n*, and
n elements in the array in sorted order, and
m, the new value to be inserted.

Output: The output list

Test cases:

#	INPUT	OUTPUT
1	n=5 LL[n]=5 10 15 20 25 m=17	Output after inserting 17: 5 10 15 17 20 25
2	n=5 LL[n]=5 10 15 20 25 m=30	Output after inserting 30: 5 10 15 20 25 30
3	n=5 LL[n]=2 4 6 8 10 m=5	Output after inserting 5: 2 4 5 6 8 10
4	n=5 LL[n]=2 4 6 8 10 m=0	Output after inserting 0: 0 2 4 6 8 10

[Time: 20 minutes]

Q4. Write a program that will delete a value from an already existing **linked list**.

User Input: Number of elements in the array, say *n*, and
n elements in inserted into the linked list, and
m, the value to be deleted
Output: The output array

Test cases:

#	INPUT	OUTPUT
1	n=5 LL[n]=5 10 15 20 25 m=15	Output after deleting 15: 5 10 20 25
2	n=5 LL[n]=5 10 15 20 25 m=30	30 does not exist
3	n=5 LL[n]=5 4 3 2 1 m=5	Output after deleting 5: 4 3 2 1
4	n=5 LL[n]=2 4 6 8 10 m=10	Output after deleting 10: 2 4 6 8

[Time: 20 minutes]

Q5. Define a structure Matrix to specify a 2D matrix.
Example:

```
typedef struct {
    int n;
    int **m;    \\ Here we will define the 2D array using malloc in main fn
} Matrix;
```

To define a 2D array dynamically you can the following code in a function:

```
void initMat(Matrix *M,int N)
{
   M->n=N;
   M->m = (int**)malloc(M->n * sizeof(int*));
   for (int i = 0; i < M->n; i++)
        M->m[i] = (int*)malloc(M->n * sizeof(int));
}
```

if you use "`M->m[i][j]`" you will get the element in the i^{th} row and j^{th} column

In main method call the above function:

```
int main()
{
   Matrix M1;
       \\ Ask user N
   initMat(&M1,N)
       \\ rest of the program
}
```

Make similar functions:

- `getMat(Matrix *M,…)`
  – input the elements taken from user into the matrix
- `printMat(Matrix *M,…)`
  – print the elements of the matrix
- `zeroMat(Matrix *M,…)`
  – Initialize a matrix with 0 as all its elements
- `multMat(Matrix* M1,Matrix* M2,Matrix* M3,…)`
  - Multiplies M1 and M2 and store in M3

Test cases:

#	INPUT	OUTPUT
1	Enter N: 2 Enter 4 numbers for M1: 1 2 3 4 Enter 4 numbers for M2: 1 0 0 1	The Matrix M1 * M2 is: 1 2 3 4
2	Enter N: 2 Enter 4 numbers for M1: 1 2 3 4 Enter 4 numbers for M2: 1 2 3 4	The Matrix M1 * M2 is: 7 10 15 22
3	Enter N: 2 Enter 4 numbers for M1: 1 1 1 1 Enter 4 numbers for M2: 1 1 1 1	The Matrix M1 * M2 is: 2 2 2 2
4	Enter N: 3 Enter 9 numbers for M1: 1 2 3 4 5 6 7 8 9 Enter 9 numbers for M2: 1 0 0 0 1 0 0 0 1	The Matrix M1 * M2 is: 1 2 3 4 5 6 7 8 9

[Time: 40 minutes]