DFS Lab

ASSIGNMENT 1

DEADLINE : 17th AUGUST

POSTED ON : 8th AUGUST

1. Programming Environment
2. Assignment Problems
   - Spiral Printing
   - Inverse Matrix
   - Modulo Arithmetic
   - Points on a Line
   - Mouse's path
   - LCD Display
   - Majority Element
   - Bonus Problem

Login to the virtual machine, where *CS17XX* is your roll number

```
ssh mtc17XX@192.168.64.35                    if you use vi or emacs
ssh -X mtc17XX@192.168.64.35                       if you use gedit
```

Set up the basic programming environment for Assignment 1

```
cd dfslab                                  go to the directory for DFS Lab
mkdir -p assign1                         create directory for Assignment 1
cd assign1                             go to the directory for Assignment 1
```

Create the solutions for Assignment 1 (e.g., in gedit)

```
gedit cs17XX-assign1-progY.c                                Y = problem number
```

First few lines of every assignment program

```
/*-------------------------------------
Name :
Roll :
Date :
Desc :
Acks :
-------------------------------------*/
```

Compiling and running your program

```
gcc -g -Wall -o progY cs17XX-assign1-progY.c
./progY
```

- A few test cases will be provided to you on the website to check the partial validity of your codes.
- The input and output should be in stdin stdout format only.
- Some border cases might be missing in the testcases. Please make sure your code works for them too.
- Don't expect us to debug your code. So please send codes without any compilation or run-time errors.

P1. Write a C program that takes as input a positive integer 'n' and generates a matrix of size $2^n x 2^n$ using pointer to pointer. Fill the matrix in a row major order with integers in the range $[1, 2^{2n}]$. Print the matrix in a spiral fashion.

Input : $n = 2$

Let the input array be of the form :

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Output :

$$1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10$$

P2. Write a C program that takes as input a positive number: row r, and allocates space dynamically for an array A of real numbers of size r x r using pointer to pointer. Take input into A from the user. Your program should compute A^{-1}, the inverse of the matrix A. Your program should also verify the correctness by multiplying A and A^{-1} to get the identity matrix.

Input : $r = 2$
Let the input array be :

$$A = \begin{bmatrix} 4 & 7 \\ 2 & 6 \end{bmatrix}$$

Output:

$$A^{-1} = \begin{bmatrix} 0.6 & -0.7 \\ -0.2 & 0.4 \end{bmatrix}$$

P3. Write a simple C program, without exploiting the power of special libraries, that takes as user input three positive integers - base x (max 32 bits), exponent n (max 32 bits), modulus m (max 32 bits) - and outputs the value of $x^n \mod m$.

Input : $x = 5,\ n = 2,\ m = 7$

Output : 4

P4. Take two initial points from the user - (x_1, y_1), (x_2, y_2) - and construct a line joining them together. Take as input a positive integer 'n', and run a loop to take 'n' more points from the user, one at a time. At the input of every new point, determine whether there was a left turn, a right turn or a straight walk in relation to the previous two points.

Input : $(0, 0)$, $(2, 2)$ and $n = 4$: $(3, 2)$, $(4, 2)$, $(5, 1)$, $(6, 4)$

Output : Right, Straight, Right, Left

P5. Write a C program to dynamically allocate an array A of size $n \times m$. n and m are inputs to be taken from the user. The array A is treated as a maze in which a mouse is trying to find its way. $A[i][j] = 0$ indicates that the mouse can step onto the location (i, j) and $A[i][j] = 1$ indicates that the mouse can't step onto the location (i, j). A mouse starts from $A[0][0]$ and has to reach $A[n-1][m-1]$, by passing only through positions that are set to 0. A mouse can move in any of the 8 directions - N, S, E, W, NE, NW, SE, SW provided that those locations have a 0 and they are valid array entries.

Input : $n = 4$ and $m = 5$
Let the input array be :

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Output : The path is

$$SE, SE, SE, E$$

or

$$SE, S, E, E, S, E$$

P6. Write a program that prints numbers in LCD display style.

Input : Digit (0 - 9) and Size (s)

Output : Print the digit in an LCD style, using s - signs for each horizontal segment and s | signs for each vertical one. Thus, each digit should occupy $s + 2$ columns and $2s + 3$ rows.
Example: Digits for $s = 2$

```
 - -
|   |       |   |       |
|   |       |   |       |
 - -         - -
|   |           |       |
|   |           |       |
 - -
```

P7. In a list of n elements, an element x is called the majority element if it appears more than $\frac{n}{2}$ times. Write a C program that takes as input from the user a list (or string) of characters, and finds the majority element in the list, if any. If there is no majority element, output NA.

Input : [A, B, B, X, B, C, D, B, F, A, B, B, L, B, B]
Output : B

Input : [A, A, B, X, B, C, D, B, F, A, B, C, L, B, B]
Output : NA

Mandar and Ansuman are playing a game picking some coins out of a pile - in turn. Each time a player is allowed to pick 1 or 2 or 4 coins, and the player that gets the last coin of the pile is the winner. Given the number of coins and the order of players (command-line), write a program to calculate the winner of the game, and calculate how many different strategies there are for that player to win the game.

```
$ ./coingame 1 ansuman mandar
ansuman 1
$ ./coingame 2 ansuman mandar
ansuamn 1
$ ./coingame 3 ansuman mandar
mandar 2
```

```
$ ./coingame 10 mandar ansuman
mandar 22
$ ./coingame 25 mandar ansuman
mandar 3344
$ ./coingame 30 mandar ansuman
ansuman 18272
```