DS Practicals

Q.1.

Output :

```
                        Practical 1

Enter the number of elements : 4

Enter the elements :
Element 1:      1
Element 2:      3
Element 3:      5
Element 4:      7

Your array is : { 1, 3, 5, 7 }

************************************************

Choose an option :

1. Linear Search (For Unordered Arrays)
2. Binary Search (For Ascending Ordered Arrays)
3. Binary Search (For Descending Ordered Arrays)

Enter your option (0 for exit) : 1

Enter the element you want to search : 5

************************************************

Your element is present in the array !!!
```

```
************************************************

Choose an option :

1. Linear Search (For Unordered Arrays)
2. Binary Search (For Ascending Ordered Arrays)
3. Binary Search (For Descending Ordered Arrays)

Enter your option (0 for exit) : 2

Enter the element you want to search : 0

************************************************

Your element is not present in the array !!!

************************************************
```

Q.2.

Output :

```
                    Practical 2

Enter the number of elements : 5

Enter the elements :
Element 1 : 4
Element 2 : 2
Element 3 : 8
Element 4 : 7
Element 5 : 1

 Your array is : { 4, 2, 8, 7, 1 }

 ********************************************************************

 Choose an option :

        1. Bubble Sort
        2. Insertion Sort
        3. Selection Sort

 Enter your option (0 for exit) : 1

 You want to sort in ?? (1 for ascending, else descending) : 1

 ********************************************************************

 Your sorted array is :

 { 1, 2, 4, 7, 8 }

 ********************************************************************
```

```
                        Practical 2

Enter the number of elements : 4

Enter the elements :
Element 1 : 8
Element 2 : 16
Element 3 : 43
Element 4 : 1

 Your array is : { 8, 16, 43, 1 }

 ********************************************************************

 Choose an option :

         1. Bubble Sort
         2. Insertion Sort
         3. Selection Sort

 Enter your option (0 for exit) : 2

 You want to sort in ?? (1 for ascending, else descending) : 2

 ********************************************************************

 Your sorted array is :

 { 43, 16, 8, 1 }

 ********************************************************************
```

```
********************************************************************

Choose an option :

        1. Bubble Sort
        2. Insertion Sort
        3. Selection Sort

Enter your option (0 for exit) : 3

You want to sort in ?? (1 for ascending, else descending) : 1

********************************************************************

Your sorted array is :

{ 1, 8, 16, 43 }

********************************************************************
```

Q.3.
Output :

```
         Practical 3

SinglyLists :
s  = {},        s1 = {12, 22, 56},       s2 = {}

Insertion of 1 to 7 in s :
s = {1, 2, 3, 4, 5, 6, 7}

Deletion of {1,4,7} in s :
s = {2, 3, 5, 6}

Reversing of s :
s = {6, 5, 3, 2}

Searching in s :
 6 in s : true
16 in s : false

Appending s1 in s :
s  = {6, 5, 3, 2, 12, 22, 56}
s1 = {12, 22, 56}

After operation s2 = s1 + s :
s  = {6, 5, 3, 2, 12, 22, 56}, s1 = {12, 22, 56},
s2 = {12, 22, 56, 6, 5, 3, 2, 12, 22, 56}
```

Q.4.

Output :

```
        Practical 4

DoublyLists :
d  = {},        d1 = {10, 20, 50}

Insertion of 1 to 5 in d :
d = {1, 2, 3, 4, 5}

Deletion of {1,4,5} in d :
d = {2, 3}

Reversing of d :
s = {3, 2}

Searching in d :
 2 in d : true
40 in d : false

Appending d in d1 :
d  = {3, 2},    d1 = {10, 20, 50, 3, 2}
```

Q.5.

Output :

```
    Practical 5

 CiruclarLists :
 d  = {},    d1 = {10, 20, 50}

 Insertion of 3 to 8 in d :
 d = {3, 4, 5, 6, 7, 8}

 Deletion of {3,4,8} in d :
 d = {5, 6, 7}

 Reversing of d :
 s = {7, 6, 5}

 Searching in d :
 5 in d : true
 0 in d : false
[Finished in 487ms]
```

Q.6.
Output :

```
        Practical 6

Stack using LinkedList

Stack details at beginning :
Stack is empty  : true
Size  of Stack  : 0

After pushing 12,7,5,9 in stack s :
Stack is empty  : false
Size  of Stack  : 4
Top of Stack    : 9

Stack s : {9, 5, 7, 12}

After poping out 9
stack s : {5, 7, 12}

Stack details at the end :
Stack is empty  : false
Size  of Stack  : 3
Top of Stack    : 5
```

Q.7.

Output :

```
       Practical 7

Stack using Arrays(with templates)

Stack details at beginning :

Stack is empty  : true
Stack is full   : false
Size  of Stack  : 0
Stack's capacity: 4

After pushing 12,7,5,9 in stack s :

Stack is empty  : false
Stack is full   : true
Size  of Stack  : 4
Stack's capacity: 4
Top of Stack    : 9

Stack s : {12, 7, 5, 9}

After poping out 9
stack s : {12, 7, 5}

Stack details at the end :

Stack is empty  : false
Stack is full   : false
Size  of Stack  : 3
Stack's capacity: 4
Top of Stack    : 5
```

Q.8.

Output :

```
        Practical 8

Queue using Circular Array(with templates)

Queue details at beginning :

Queue is empty  : true
Queue is full   : false
Size  of Queue  : 0
Queue's capacity: 4

After enqueuing 1 to 4 in Queue :

Queue is empty  : false
Queue is full   : true
Size  of Queue  : 4
Queue's capacity: 4
Front of Queue  : 1
Back  of Queue  : 4

Queue : {1, 2, 3, 4}

After dequeuing 1
Queue : {2, 3, 4}

Queue details at the end :

Queue is empty  : false
Queue is full   : false
Size  of Queue  : 3
Queue's capacity: 4
Front of Queue  : 2
Back  of Queue  : 4
```

Q.9.

Output :

```
        Practical 9

Deque using Linked List

Deque details at beginning :

Deque is empty  : true
Size  of Deque  : 0


After enqueuing(at front) 1 to 4 in Deque :
Deque : {4, 3, 2, 1}

After pushing(at end) 5 to 8 in Deque :
Deque : {4, 3, 2, 1, 5, 6, 7, 8}

After popping out from the front :
Deque : {3, 2, 1, 5, 6, 7, 8}

After popping out from the end :
Deque : {3, 2, 1, 5, 6, 7}

Deque details at the end :

Deque is empty  : false
Size  of Deque  : 6
Front of Deque  : 3
Back  of Deque  : 7
```

Q.10.

Output :

```
        Practical 10

Enter the highest degree of polynomials : 2

First Polynomial's Parameter :
Enter the coefficients (descending order) : 4 3 2

Second Polynomial's Parameter :
Enter the coefficients (descending order) : 1 5 9

First Polynomial : 4x^2 + 3x + 2

Second Polynomial : 1x^2 + 5x + 9

After addition of these polymonials :
Final Polynomial  : 5x^2 + 8x + 11
```

Q.11.

Output :

```
        Practical 11
  Factorial & Factor calculator

 Enter a number :  10

 From which process you want to calculate  :
 1. Using recursion
 2. Using iterator (loops)

 Enter your choice :  1


        Using Recursion....

Factor of 10    : 1     2       5       10
Factorial of 10  : 3628800
```

```
        Practical 11
  Factorial & Factor calculator

 Enter a number :  7

 From which process you want to calculate  :
 1. Using recursion
 2. Using iterator (loops)

 Enter your choice :  2


        Using Iterator (Loops)
Factor of 7     : 1     7
Factorial of 7  : 5040
```

Q.12.

Output :

```
                        Practical 12
                Fibonacci Series Displayer

Enter the term till you want to display  :  5

What process would you like :
1. Using recursion
2. Using iteration

Enter your choice :  1

Fibonacci Series till nth term is :

 0      1       1       2       3
```

```
                    Practical 12
               Fibonacci Series Displayer

Enter the term till you want to display  :  8

What process would you like :
1. Using recursion
2. Using iteration

Enter your choice :  2

Fibonacci Series till nth term is :

 0     1       1       2       3       5       8       13
```

Q.13.

Output :

```
        Practical 13

GCD CALCULATOR

Enter two numbers (use space):  5 25

Which process do you like :

        1. With Recursion
        2. Without Recursion

Enter your choice : 1

GCD (5,25) = 5
```

```
        Practical 13

GCD CALCULATOR

Enter two numbers (use space):  7 8

Which process do you like :

        1. With Recursion
        2. Without Recursion

Enter your choice : 2

GCD (7 , 8) = 1
```

Q.14.

Output :

```
    Practical 14

Binary Search Trees

Inserting 8,1 in Tree using recursion :
b = 8  1
Inserting 10,9,11,2,5 in Tree using iteration :
b = 8  1   2   5   10  9   11

Deletion of 9 via merging :
b = 8  1   2   5   10  11
Deletion of 11 by copying :
b = 8  1   2   5   10

Height of tree   : 3
Total Node Count : 5
Leaf Node Count  : 2
Non Leaf Count   : 3
```

```
Traversal of Binary Search Tree b :

Iterative Version :

Inorder Traversal     : 1  2   5   8   10
Pre-order Traversal   : 8  1   2   5   10
Post-order Traversal  : 5  2   1   10  8
Level-order Traversal : 8  1   10  2   5

Recursive Version :

Inorder Traversal     : 1  2   5   8   10
Pre-order Traversal   : 8  1   2   5   10
Post-order Traversal  : 5  2   1   10  8
Level-order Traversal : 8  1   10  2   5

Binary Search Tree b    : 1    2   5   8   10
Mirror of Binary Tree b : 10   8   5   2   1

Binary Search Trees :
b  : 1 2   5   8   10
b1 : 5 7   9
(b == b1)  : false
```

Q.15.

Output :

```
 Practical 15

Sparse Matrix :
0   0   0   0   9   0
0   8   0   0   0   0
4   0   0   2   0   0
0   0   0   0   0   5
0   0   2   0   0   0


Non-Zero Representation :
0 1 2 2 3 4
4 1 0 3 5 2
9 8 4 2 5 2
[Finished in 454ms]
```

Q.16.

Output :

```
avinash@avinash-OptiPlex-390:/media/avinash/F012C43312C40098/B.Sc_CS/#Seme

        Stack  size  capacity   Data

         s1 :   10      10      {2, 3, 7, 9, 11, 15, 17, 120, 125, 150}
         s2 :   0       10      {}

After swapping/moving

         s1 :   0       10      {}
         s2 :   10      10      {2, 3, 7, 9, 11, 15, 17, 120, 125, 150}

After reversing

         s1 :   10      10      {150, 125, 120, 17, 15, 11, 9, 7, 3, 2}
         s2 :   0       10      {}
```

Q.17.

Output :

```
avinash@avinash-OptiPlex-390:/media/avinash/F012C43312C40098/B.Sc_CS/#Seme

ADT = STACK(s1) / Queue(s2)

         ADT    size  capacity  Data

         s1 :   10      10      {2, 3, 7, 9, 11, 15, 17, 120, 125, 150}
         s2 :   0       10      {}

After swapping/moving

         s1 :   0       10      {}
         s2 :   10      10      {150, 125, 120, 17, 15, 11, 9, 7, 3, 2}

After reversing

         s1 :   10      10      {150, 125, 120, 17, 15, 11, 9, 7, 3, 2}
         s2 :   0       10      {}
avinash@avinash-OptiPlex-390:/media/avinash/F012C43312C40098/B.Sc_CS/#Seme
```

Q.18.

Output :

```
        Practical 18
Diagonal Matrix representation in 1D Array

Enter the row & column of the matrix : 2 2

Enter the 2 diagonal elements : 5 8

Diagonal Matrix :
       5       0
       0       8
```

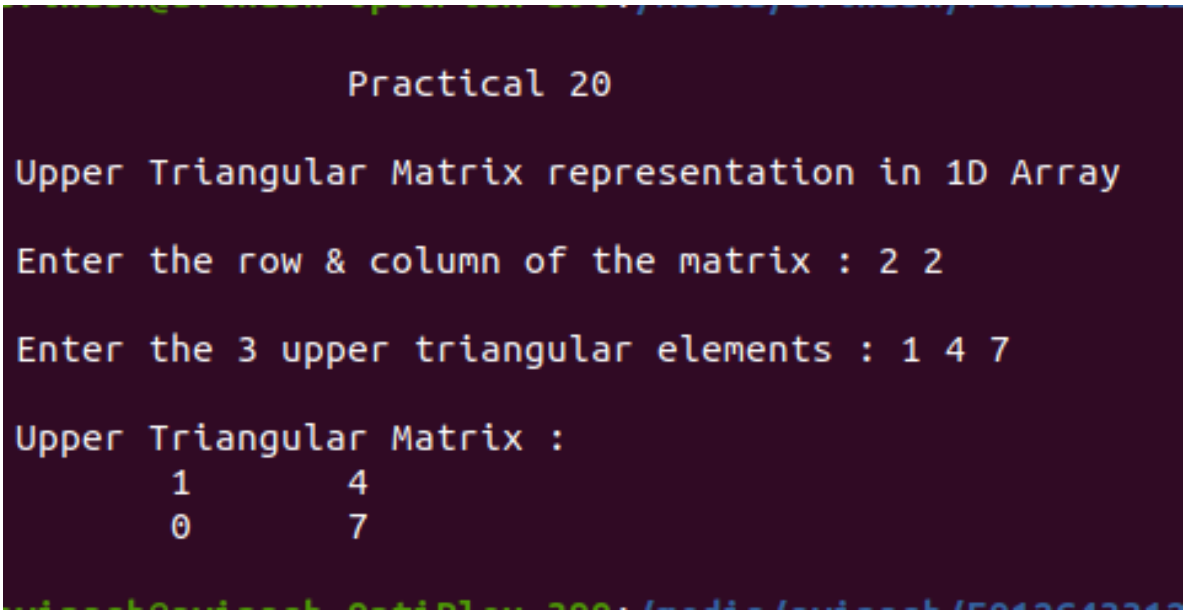
Q.19.

Output :

```
              Practical 19

Lower Triangular Matrix representation in 1D Array

Enter the row & column of the matrix : 3 3

Enter the 6 lower triangular elements : 1 3 4 5 6 7

Lower Triangular Matrix :
       1       0       0
       3       4       0
       5       6       7
```

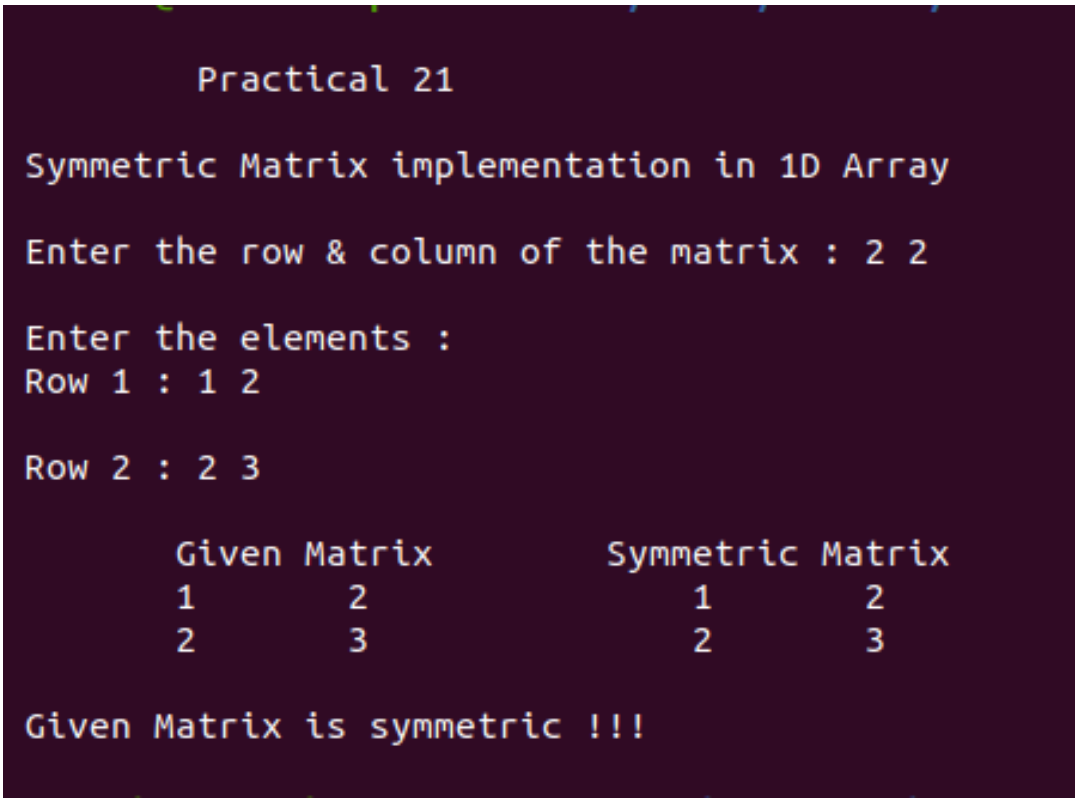
Q.20.

Output :



Q.21.

Output :



Q.23.

Output :

```
Practical 23

 AVL Tree

 Insertion in AVL :

 After inserting 5 in AVL :
 5

 After inserting 7 in AVL :
 5
 7

 After inserting 8 in AVL :
 7
 5 8

 After inserting 6 in AVL :
 7
 5 8
 6

 After deleting 8 in AVL :
 6
 5 7
[Finished in 698ms]
```

Q.24.

Output :

```
   Practical 24

Heap (with templates)

Heap details at beginning :

Heap is empty  : true
Heap is full   : false
Size  of Heap  : 0
Heap's capacity: 10

After pushing 12,7,5,9 in Heap s :

Heap is empty  : false
Heap is full   : false
Size  of Heap  : 4
Heap's capacity: 10
Top of Heap    : 5

Heap s : {5, 9, 7, 12}

After poping out the Top5
Heap s : {7, 9, 12}

Heap details at the end :

Heap is empty  : false
Heap is full   : false
Size  of Heap  : 3
Heap's capacity: 10
Top of Heap    : 7
[Finished in 478ms]
```