Name - Shivam Gupta
Roll No.: 21095106
Discipline - ECE

Q1. Write a program to reverse an array or string
Sol ->

```cpp
#include <bits/stdc++.h>
using namespace std;

int main()
{
   int n;
   cout << "Enter number of elements: ";
   cin >> n;
   // input elements of array
   char arr[n][1000];
   for (int i = 0; i < n; i++)
   {
       cin >> arr[i];
   }
   cout << "Given array:" << endl;
   for (int i = 0; i < n; i++)
   {
       cout << arr[i] << " ";
   }
   cout << endl;
   // reverse array
   char temp[1000];
   for (int i = 0, j = n - 1; i < j; i++, j--)
   {
       strcpy(temp, arr[i]);
       strcpy(arr[i], arr[j]);
       strcpy(arr[j], temp);
   }

   // print reversed array
   cout << "Reversed array:" << endl;
   for (int i = 0; i < n; i++)
   {
       cout << arr[i] << " ";
   }
   printf("\n");

   return 0;
```

```
}
```

Output :-

```
➜  assignment1 git:(master) ✗ g++ arrayReverse.cpp
➜  assignment1 git:(master) ✗ ./a.out
Enter number of elements: 5
1 2 3 4 5
Given array:
1 2 3 4 5
Reversed array:
5 4 3 2 1
➜  assignment1 git:(master) ✗ ./a.out
Enter number of elements: 6
1 5 87 2
+ -
Given array:
1 5 87 2 + -
Reversed array:
- + 2 87 5 1
```

Q2. Write a program to reverse the single linked list and double linked list.
Sol=>

(1) Single Linked List

```
#include<bits/stdc++.h>
using namespace std;

struct Node
{
    int data;
    struct Node *next;
    Node(int data)
    {
        this->data = data;
        next = NULL;
    }
};

struct LinkedList
{
    Node *head;
    LinkedList() { head = NULL; }

    // Function to reverse the linked list
    void reverse()
    {
        Node *current = head;
        Node *prev = NULL, *next = NULL;

        while (current != NULL)
        {
            next = current->next;
            current->next = prev;
            prev = current;
            current = next;
        }
        head = prev;
    }

    // Function to print linked list
    void print()
    {
        struct Node *temp = head;
        while (temp != NULL)
        {
            cout << temp->data << " ";
```

```
            temp = temp->next;
        }
    }

    // Function to input data
    void push(int data)
    {
        Node *temp = new Node(data);
        temp->next = head;
        head = temp;
    }
};

int main()
{
    LinkedList ll;
    ll.push(1);
    ll.push(2);
    ll.push(3);
    ll.push(4);

    cout << "Given linked list:" << endl;
    ll.print();

    ll.reverse();

    cout << "\nReversed Linked list: " << endl;
    ll.print();
    return 0;
}
```

Output:-

```
➜  assignment1 git:(master) ✗ g++ sllReverse.cpp
➜  assignment1 git:(master) ✗ ./a.out
Given linked list:
4 3 2 1
Reversed Linked list:
1 2 3 4 %
```

(2) Double Linked List

```
#include <bits/stdc++.h>
using namespace std;

class Node
{
public:
   int data;
   Node *next;
   Node *prev;
};

/* Function to reverse a Doubly Linked List */
void reverse(Node **head_ref)
{
   Node *temp = NULL;
   Node *current = *head_ref;

   while (current != NULL)
   {
       temp = current->prev;
       current->prev = current->next;
       current->next = temp;
       current = current->prev;
   }

   if (temp != NULL)
       *head_ref = temp->prev;
}


void push(Node **head_ref, int new_data)
{
   Node *new_node = new Node();

   new_node->data = new_data;

   new_node->prev = NULL;

   new_node->next = (*head_ref);

   if ((*head_ref) != NULL)
       (*head_ref)->prev = new_node;

   (*head_ref) = new_node;
```

```
}

void printList(Node *node)
{
   while (node != NULL)
   {
       cout << node->data << " ";
       node = node->next;
   }
}

int main()
{
   Node *head = NULL;

   push(&head, 1);
   push(&head, 2);
   push(&head, 3);
   push(&head, 4);

   cout << "Given Linked list:" << endl;
   printList(head);

   reverse(&head);

   cout << "\nReversed Linked list:" << endl;
   printList(head);

   return 0;
}
```

Output ->

```
→  assignment1 git:(master) ✗ g++ dllReverse.cpp
→  assignment1 git:(master) ✗ ./a.out
Given Linked list:
4 3 2 1
Reversed Linked list:
1 2 3 4 %
```

Q4. Write a program to implement push & pop operations in stack using array and linked list.

Sol=>

(1) Array

```
#include <bits/stdc++.h>
using namespace std;

class Stack
{
    int top;

public:
    int a[1000];
    Stack() { top = -1; }
    bool push(int x);
    int pop();
    int peek();
    bool isEmpty();
};

bool Stack::push(int x)
{
    if (top >= (999))
    {
        cout << "Stack Overflow";
        return false;
    }
    else
    {
        a[++top] = x;
        cout << x << " pushed into stack\n";
        return true;
    }
}

int Stack::pop()
{
    if (top < 0)
    {
        cout << "Stack Underflow";
        return 0;
    }
    else
    {
        int x = a[top--];
```

```
        return x;
    }
}
int Stack::peek()
{
    if (top < 0)
    {
        cout << "Stack is Empty";
        return 0;
    }
    else
    {
        int x = a[top];
        return x;
    }
}

bool Stack::isEmpty()
{
    return (top < 0);
}

int main()
{
    class Stack s;
    s.push(1);
    s.push(2);
    s.push(3);
    cout << s.pop() << " Popped from stack\n";
    cout << "Elements present in stack : ";
    while (!s.isEmpty())
    {
        cout << s.peek() << " ";
        s.pop();
    }
    return 0;
}
```

Output-

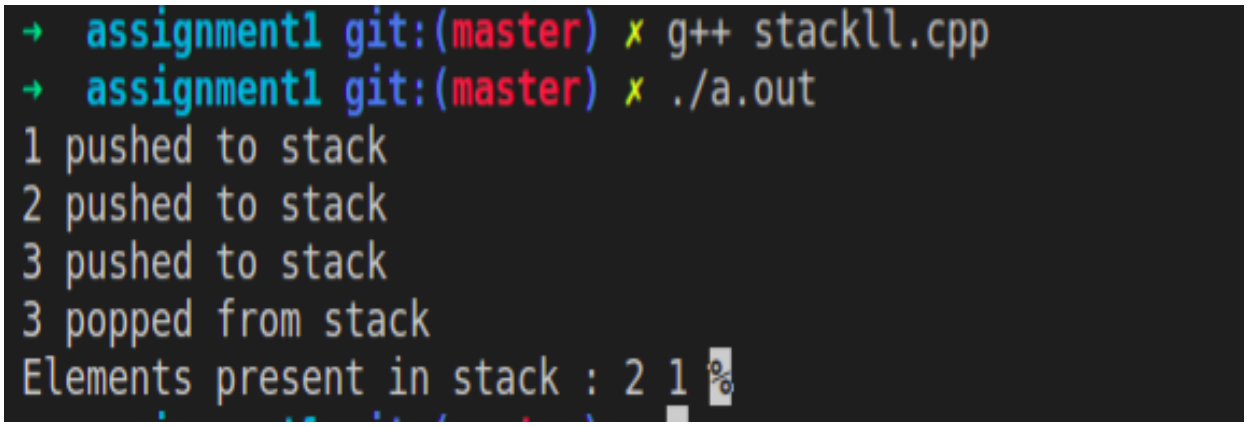
```
→  assignment1 git:(master) ✗ g++ stackArr.cpp
→  assignment1 git:(master) ✗ ./a.out
1 pushed into stack
2 pushed into stack
3 pushed into stack
3 Popped from stack
Elements present in stack : 2 1 %
```

(2) Linked List

```
#include <bits/stdc++.h>
using namespace std;

class StackNode
{
public:
    int data;
    StackNode *next;
};

StackNode *newNode(int data)
{
    StackNode *stackNode = new StackNode();
    stackNode->data = data;
    stackNode->next = NULL;
    return stackNode;
}

int isEmpty(StackNode *root)
{
    return !root;
}

void push(StackNode **root, int data)
{
    StackNode *stackNode = newNode(data);
    stackNode->next = *root;
    *root = stackNode;
    cout << data << " pushed to stack\n";
}

int pop(StackNode **root)
{
    if (isEmpty(*root))
        return INT_MIN;
    StackNode *temp = *root;
    *root = (*root)->next;
    int popped = temp->data;
    free(temp);

    return popped;
}
```

```
int peek(StackNode *root)
{
   if (isEmpty(root))
       return INT_MIN;
   return root->data;
}

int main()
{
   StackNode *root = NULL;

   push(&root, 1);
   push(&root, 2);
   push(&root, 3);

   cout << pop(&root) << " popped from stack\n";

   cout << "Elements present in stack : ";
   while (!isEmpty(root))
   {
       cout << peek(root) << " ";
       pop(&root);
   }

   return 0;
}
```

Output:-



Q5. Write a program for Tower of Hanoi
Sol:-

```cpp
#include <bits/stdc++.h>
using namespace std;

void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod)
{
    if (n == 0)
    {
        return;
    }
    towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
    cout << "Move disk " << n << " from rod " << from_rod << " to rod " << 
to_rod << endl;
    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}

int main()
{
    int n;
    cout<<"Enter number of disks: ";
    cin >> n;
    towerOfHanoi(n, 'A', 'C', 'B');
    int c = pow(2,n);
    cout<<endl<<"Total Number of moves = "<<c<<endl;
    return 0;
}
```

Output -

```
→  assignment1 git:(master) ✗ g++ towerHanoi.cpp
→  assignment1 git:(master) ✗ ./a.out
Enter number of disks: 3
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C

Total Number of moves = 8
```

Q6. Write a program to insertion and deletion in Queue.
Sol->

```
#include <bits/stdc++.h>
using namespace std;

struct Queue
{
   int front, rear, capacity;
   int *queue;
   Queue(int c)
   {
       front = rear = 0;
       capacity = c;
       queue = new int;
   }

   ~Queue() { delete[] queue; }

   void queueEnqueue(int data)
   {
       if (capacity == rear)
       {
           cout << "\nQueue is full\n";
           return;
       }
       else
       {
           queue[rear] = data;
           rear++;
       }
       return;
   }

   void queueDequeue()
   {
       if (front == rear)
       {
           cout << "\nQueue is empty\n";
           return;
       }
       else
       {
           for (int i = 0; i < rear - 1; i++)
           {
               queue[i] = queue[i + 1];
```

```
            }
            rear--;
        }
        return;
    }

    void queueDisplay()
    {
        int i;
        if (front == rear)
        {
            cout << "\nQueue is Empty\n";
            return;
        }
        for (i = front; i < rear; i++)
        {
            cout << queue[i] << "  <-- ";
        }
        return;
    }
};

int main(void)
{
    cout << "Queue is Empty" << endl;
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    Queue q(n);

    for (int i = 0; i < n; i++)
    {
        int x;
        cin >> x;
        q.queueEnqueue(x);
    }
    q.queueDisplay();
    cout << endl;

    q.queueDequeue();
    cout << endl << "After node deletion" << endl;
    q.queueDisplay();
    cout<<endl;
```

```
    cout << "Enter element to insert at end: ";
    int ins;
    cin >> ins;
    q.queueEnqueue(ins);
    q.queueDisplay();

    return 0;
}
```

Output:-

```
→  assignment1 git:(master) ✗ g++ queue.cpp
→  assignment1 git:(master) ✗ ./a.out
Queue is Empty
Enter number of elements: 4
1 2 3 5
1  <-- 2  <-- 3  <-- 5  <--

After node deletion
2  <-- 3  <-- 5  <--
Enter element to insert at end: 4
2  <-- 3  <-- 5  <-- 4  <--
```