STACK USING SINGLY LINKED LIST :

```
#include<iostream>

using namespace std;

class Node {
  public:
    int key;
  int data;
  Node * next;

  Node() {
    key = 0;
    data = 0;
    next = NULL;
  }
  Node(int k, int d) {
    key = k;
    data = d;
    next = NULL;
  }
};

class Stack {
  public:
    Node * top;

  Stack() {
    top = NULL;
  }

  bool isEmpty() {
    if (top == NULL) {
      return true;
    } else {
      return false;
    }
  }
  bool checkIfNodeExist(Node * n) {
    Node * temp = top;
    bool exist = false;
    while (temp != NULL) {
      if (temp - > key == n - > key) {
        exist = true;
        break;
      }
      temp = temp - > next;
    }
    return exist;
  }

  void push(Node * n) {
```

```
    if (top == NULL) {
      top = n;
      cout << "Node  PUSHED successfully" << endl;
    } else if (checkIfNodeExist(n)) {
      cout << "Node already exist with this Key value." <<
        "Enter different Key value" << endl;
    } else {
      Node * temp = top;
      top = n;
      n - > next = temp;
      cout << "Node  PUSHED successfully" << endl;
    }

  }

  Node * pop() {
    Node * temp = NULL;
    if (isEmpty()) {
      cout << "stack underflow" << endl;
      return temp;
    } else {
      temp = top;
      top = top - > next;
      return temp;
    }
  }
  Node * peek() {
    //Node *temp = NULL;
    if (isEmpty()) {
      cout << "stack underflow" << endl;
      return NULL;
    } else {
      return top;
    }
  }
  int count() {
    int count = 0;
    Node * temp = top;
    while (temp != NULL) {
      count++;
      temp = temp - > next;
    }
    return count;
  }
  void display() {
    cout << "All values in the Stack are :" << endl;
    Node * temp = top;
    while (temp != NULL) {
      cout << "(" << temp - > key << "," << temp - > data << ")" <<
" -> " <<
        endl;
      temp = temp - > next;
    }
    //cout<< "Total no of Nodes in the stack :" <<count()
```

```
    //<<endl;
    cout << endl;
  }

};

int main() {
  Stack s1;
  int option, key, data;

  do {
    cout << "What operation do you want to perform?" <<
      "Select Option number. Enter 0 to exit." << endl;
    cout << "1. Push()" << endl;
    cout << "2. Pop()" << endl;
    cout << "3. isEmpty()" << endl;
    cout << "4. peek()" << endl;
    cout << "5. count()" << endl;
    cout << "6. display()" << endl;
    cout << "7. Clear Screen" << endl << endl;
    cin >> option;

    //Node n1 = new Node();
    Node * new_node = new Node();

    switch (option) {
    case 0:
      break;
    case 1:
      cout << "Enter KEY and VALUE of NODE to push in the stack" <<
        endl;
      cin >> key;
      cin >> data;
      new_node - > key = key;
      new_node - > data = data;
      s1.push(new_node);
      break;
    case 2:
      cout << "Pop Function Called - Poped Value: " << endl;
      new_node = s1.pop();
      cout << "TOP of Stack is: (" << new_node - > key << "," <<
new_node - > data << ")";
      delete new_node;
      cout << endl;

      break;
    case 3:
      if (s1.isEmpty())
        cout << "Stack is Empty" << endl;
      else
        cout << "Stack is not Empty" << endl;
      break;
    case 4:
      if (s1.isEmpty()) {
```

```
        cout << "Stack is Empty" << endl;
      } else {
        cout << "PEEK Function Called : " << endl;
        new_node = s1.peek();
        cout << "TOP of Stack is: (" << new_node - > key << "," <<
new_node - > data << ")" <<
          endl;
      }
      break;
    case 5:
      cout << "Count Function Called: " << endl;
      cout << "No of nodes in the Stack: " << s1.count() << endl;
      break;

    case 6:
      cout << "Display Function Called - " << endl;
      s1.display();
      cout << endl;
      break;
    case 7:
      system("cls");
      break;
    default:
      cout << "Enter Proper Option number " << endl;
    }

  } while (option != 0);

  return 0;
}
```