# TASK -1

#include <iostream>

using namespace std;

class Stack

{

private:

    struct Node

    {

        int data;

        Node \*next;

    };

    Node \*top;

public:

    Stack()

    {

        top = NULL;

    }

    bool push(int x)

    {

        Node \*temp = new Node;

        if (temp == NULL)

            return false;

        temp->data = x;

        temp->next = top;

        top = temp;

        return true;

    }

    int pop()

    {

        int x = -1;

        if (top == NULL)

            return x;

        Node \*temp = top;

        top = top->next;

        x = temp->data;

        delete temp;

        return x;

    }

    bool isEmpty()

    {

        return top == NULL;

    }

    void display()

    {

        Node \*temp = top;

        while (temp != NULL)

        {

            cout << temp->data << " ";

            temp = temp->next;

        }

        cout << endl;

    }

};

int evaluate(int a, int b, char op)

{

    switch (op)

    {

    case '+':

        return a + b;

    case '-':

        return a - b;

    case '\*':

        return a \* b;

    case '/':

        return a / b;

    }

}

int main()

{

    Stack s;

    string exp;

    cout << "Enter an expression: ";

    cin >> exp;

    int i = 0;

    while (exp[i] != '\0')

    {

        if (exp[i] == ' ')

            i++;

        else if (isdigit(exp[i]))

        {

            int val = 0;

            while (i < exp.length() &&

                   isdigit(exp[i]))

            {

                val = (val \* 10) + (exp[i] - '0');

                i++;

            }

            i--;

            s.push(val);

        }

        else

        {

            int val1 = s.pop();

            int val2 = s.pop();

            switch (exp[i])

            {

            case '+':

                s.push(val2 + val1);

                break;

            case '-':

                s.push(val2 - val1);

                break;

            case '\*':

                s.push(val2 \* val1);

                break;

            case '/':

                s.push(val2 / val1);

                break;

            }

        }

        i++;

    }

    cout << "Result: " << s.pop();

    return 0;

}

# Task – 2

#include <iostream>

#include <stack>

using namespace std;

int main()

{

    string s, word = "";

    stack<string> stc;

    getline(cin, s); // take the entire sentence as input

    for (int i = 0; i <= s.length(); i++)

    {

        if (s[i] == ' ' or s[i] == '\0') /\*if it is end of string or a blank space the push the word to the stack\*/

        {

            stc.push(word);

            word = "";

        }

        else if (isalnum(s[i])) /\*this ensures that only alphanumeric characters are taken as part of words\*/

        {

            word = word + s[i];

        }

    }

    int sz = stc.size();

    while (!stc.empty())

    {

        if (sz == stc.size())

            stc.top()[0] -= 32; // First letter of first should be capital

        if (stc.size() == 1)

            stc.top()[0] += 32;

        cout << stc.top() << " ";

        stc.pop();

    }

}

# Task – 3

#include <bits/stdc++.h>

using namespace std;

typedef struct Stack

{

   int size;

   int top;

   int \*arr;

}Stack\_Node;

Stack\_Node\* Create\_stack(int size)

    {

        Stack\_Node\* stack= new Stack;

        stack->size =size;

        stack->top =-1;

        stack->arr = new int[size];

        return stack;

    }

// Function to check stack is full or not

bool Check\_Full(Stack\_Node\* stack)

    {

       if(stack->top == stack->size - 1)

          return true;

       else

         return false;

     }

// Function to check stack is empty or not

bool Check\_Empty(Stack\_Node\* stack)

    {

       if(stack->top == -1)

         return true;

      else

        return false;

    }

// Function to Push the element in stack

void push(Stack\_Node\* stack, int element)

   {

      if(Check\_Full(stack))

         return;

      stack->arr[++stack->top]= element;

   }

// Function to Pop the element and return Popped element

int pop(Stack\_Node\* stack)

   {

        if(Check\_Empty(stack))

           return -1;

        return(stack->arr[stack->top--]);

    }

// Function to Print the Movement of Discs

void Move\_Disc(int disc, char from\_Rod, char to\_Rod)

   {

       cout<<"Move the disc "<<disc<<" "<<"from Rod '"<<from\_Rod<<"' to Rod '"<<to\_Rod<<"'"<<endl;

   }

void Move\_Disc\_Helper(struct Stack \*source, struct Stack\*dest, char s, char d)

   {

      int top1=pop(source);

      int top2 =pop(dest);

      if (top1 == -1)

       {

           push(source,top2);

           Move\_Disc(top2, d, s);

       }

      else if (top2 ==-1)

      {

              push( dest,top1);

              Move\_Disc(top1, s, d);

      }

      else if (top1 >top2)

      {

              push(source, top1);

              push(source, top2);

              Move\_Disc(top2, d, s);

       }

      else

       {

             push( dest,top2);

             push( dest,top1);

             Move\_Disc(top1, s, d);

       }

 }

void TowerOfHanoi(int number\_of\_discs, struct Stack\*source, struct Stack \*aux, struct Stack \*dest)

    {

            char s = 'S', d = 'D',a = 'A';

            //  if n is even swap aux and dest

            if (number\_of\_discs % 2== 0)

            {

                        char var =d;

                        d =a;

                        a =var;

            }

            int number\_of\_moves =pow(2, number\_of\_discs) - 1;

            for (int i =number\_of\_discs; i >= 1; i--)

            {

                push(source,i);

            }

            // iteration of each i upto number of moves

            for (int i = 1; i <=number\_of\_moves; i++)

            {

                if (i % 3== 0)

                      Move\_Disc\_Helper(aux, dest, a, d);

               else if (i% 3 == 2)

                      Move\_Disc\_Helper(source,aux, s, a);

               else if (i% 3 == 1)

                      Move\_Disc\_Helper(source,dest, s, d);

            }

    }

    int main()

    {

            int number\_of\_discs;

            cin>>number\_of\_discs;

            Stack\_Node\* source;

            Stack\_Node\* dest;

            Stack\_Node\* aux;

            // Creating 3 stacks for the three Rods

            source =Create\_stack(number\_of\_discs);

            aux =Create\_stack(number\_of\_discs);

            dest =Create\_stack(number\_of\_discs);

            TowerOfHanoi(number\_of\_discs,source, aux, dest);

            // delete dynamically allocated memory stacks

            delete source;

            delete aux;

            delete dest;

            return 0;

  }

# Task - 4

#include<bits/stdc++.h>

using namespace std;

class traffic{

    deque<int> north;

    deque<int> south;

    public:

    void addCar(int dir){

        if(dir == 1){

            north.push\_back(1);

        }

        else if(dir == 2)

        {

            south.push\_back(1);

        }

    }

    void removeCar(int dir)

    {

        if(dir == 1){

            north.pop\_back();

        }

        else if(dir == 2){

            south.pop\_back();

        }

        if(north.size() == 0 && south.size() == 0)

        {

            cout<<"Traffic is clear";

        }

    }

};

# Task- 4b

(A.) Since 2015, a majority of Since 2015, a majority of American have had unfavourable View of the ACA - This statement is FALSE,

(B) Provision draws the most support among Demeentes, Independetes and Repulicans

\* Allours young insurance plans. adults to stay untill age 26" on their parents"

Eliminates Out. of Pocket costs for many prauntive fervices \* Creates Health & insurance exchanges where small

buisnesses and people can shop for and Compare Coverage.

(C) ACA Provision on which democrats and Repulicans the most at odds on are

\* Requires nearly all Americans to have health insurance Or else Pay a fine & D.

(1) Requires employers with 10+ employers to if they don't offer health Insurance рау a fine.

# TASK – 5

# TASK – 6

#include <iostream>

**using** **namespace** std;

**struct** node {

**int** data;

**struct** node \*next;

};

**struct** node\* front = NULL;

**struct** node\* rear = NULL;

**struct** node\* temp;

**void** Insert(**int** val) {

**if** (rear == NULL) {

rear = **new** node;

rear->next = NULL;

rear->data = val;

front = rear;

     } **else** {

temp=**new** node;

rear->next = temp;

temp->data = val;

temp->next = NULL;

rear = temp;

   }

}

**void** Delete() {

temp = front;

**if** (front == NULL) {

cout<<"Queue is empty!!"<<endl; } **else** **if** (temp->next != NULL) {

temp = temp->next;

cout<<"Element deleted from queue is : "<<front->data<<endl;

**free**(front);

front = temp;

   } **else** {

cout<<"Element deleted from queue is : "<<front->data<<endl;

**free**(front);

front = NULL;

rear = NULL;

   }

}

**void** Display() {

temp = front;

**if** ((front == NULL) && (rear == NULL)) {

cout<<"Queue is empty"<<endl;

**return**;

   }

**while** (temp != NULL) {

cout<<temp->data<<" "; temp = temp->next;

   }

cout<<endl;

}

**int** main() {

cout<<"Queue Created:"<<endl;

Insert(10);

Insert(20);

Insert(30);

Insert(40);

Insert(50);

Display();

Delete();

cout<<"Queue after one deletion: "<<endl;

Display();

**return** 0;

}