Stack Assignment No. = 1

1) Write a program to implement stack by using array. (Static Implementation of stack)
-->

```
#include<iostream.h>
#include<conio.h>

class stack
{
  int *items, top, size;
  public:
     void    create(stack*, int);
     void    push(stack*, int);
     int     pop(stack*);
     void    isempty(stack*);
     void    isfull(stack*);
     void    display(stack*);
};
void     stack::create(stack *p, int size)
{
       p-> top = -1;
       p-> items = new int[size];
       p-> size = size;
       cout << "\nStack is created.." << endl;
}
void    stack::push(stack *p, int ele)
{
        if(p->top == p->size-1) {
             cout << "\nStack overflows.." << endl;
        }
        else {
             p-> items[++p->top] = ele;
             cout << "\nElement Pushed" << endl;
        }
}

int    stack::pop(stack *p)
{
        if (p-> top == -1) {
             cout << "\nStack underflows.." << endl;
             return 0;
        }
        else {
             return p-> items[p-> top--];
        }
}

void     stack::isfull(stack *p)
{
        if (p->top == p-> size-1){
             cout << "\nStack is FULL" << endl;
        }
        else {
             cout << "\nStack is NOT FULL" << endl;
        }
}
void    stack::isempty(stack *p)
```

```
{
        if (p-> top == -1){
             cout << "\nStack is EMPTY" << endl;
        }
        else {
             cout << "\nStack is NOT EMPTY" << endl;
        }
}
void   stack::display(stack *p)
{
        cout << "\nElements: ";
        for (int i = p->top; i >= 0; i--) {
             cout << p->items[i] << " ";
        }

        if (p-> top == -1){
            cout << "0" << endl;
        }
}
void  main()
{
    clrscr();
    stack  obj,q;
    stack  *p = &q;
    int  ch, size, ele;
    do
    {
        cout << "\n1 : Create\n2 : Isfull\n3 : Isempty\n4 : Push\n5 :
Pop\n6 : Dispaly\n7 : Exit\nEnter your choice: ";
        cin >> ch;
        switch (ch)
        {
                case 1:
                    cout << "\nEnter the size of stack: ";
                    cin >> size;
                    obj.create(p, size);
                    break;
                case 2:
                    obj.isfull(p);
                    break;
                case 3:
                    obj.isempty(p);
                    break;
                case 4:
                    cout << "\nEnter element to push: ";
                    cin >> ele;
                    obj.push(p, ele);
                    break;
                case 5:
                     cout << "\nPopped element: " << obj.pop(p) << endl;
                     break;
                case 6:
                    obj.display(p);
                    break;
        }
    } while (ch != 7);

    getch();
}
```

```
Output -->

1 : Create
2 : Isfull
3 : Isempty
4 : Push
5 : Pop
6 : Dispaly
7 : Exit
Enter your choice: 1

Enter the size of stack: 2

Stack is created..

1 : Create
2 : Isfull
3 : Isempty
4 : Push
5 : Pop
6 : Dispaly
7 : Exit
Enter your choice: 2

Stack is NOT FULL

1 : Create
2 : Isfull
3 : Isempty
4 : Push
5 : Pop
6 : Dispaly
7 : Exit
Enter your choice: 3

Stack is EMPTY

1 : Create
2 : Isfull
3 : Isempty
4 : Push
5 : Pop
6 : Dispaly
7 : Exit
Enter your choice: 4

Enter element to push: 100

Element Pushed

1 : Create
2 : Isfull
3 : Isempty
4 : Push
5 : Pop
6 : Dispaly
7 : Exit
Enter your choice: 6

Elements: 100
```

```
1 : Create
2 : Isfull
3 : Isempty
4 : Push
5 : Pop
6 : Dispaly
7 : Exit
Enter your choice: 7
```

2) Write a program, which reverses the entered string by using stack
-->

```
#include<iostream.h>
#include<conio.h>
#define  max  50

class   stack
{
    public:
       int items[max], top;
       void    create(stack*);
       void    push(stack*, int);
       int     pop(stack*);
};

void   stack::create(stack *p)
{
       p-> top = -1;
}
void   stack::push(stack *p, int ele)
{
       if (p-> top == max-1) {
            cout << "\nStack overflows.." << endl;
       }
       else {
            p-> items[++p-> top] = ele;
       }
}
int    stack::pop(stack *p)
{
       return   (p-> top == -1)? 0 : p-> items[p-> top--];
}
void  main()
{
       clrscr();
       char  str[max], rev[max];
       int   i = 0, j = 0;
       stack  obj, q;
       stack  *p = &q;

       obj.create(p);

       cout << "Enter any string : ";
       cin >> str;

       while (str[i] != '\0')
       {
             obj.push(p, str[i]);
```

```
              i++;
       }

       while (p-> top != -1)
       {
              rev[j++] = obj.pop(p);
       }
       rev[j] = '\0';

       cout << "Reversed string : " << rev << endl;
       getch();
}

Output -->
Enter any string : Shreyash
Reversed string : hsayerhS
```

3) Write a program to check entered string is palindrome or not by using stack
-->

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#define  max  50

class   stack
{
    public:
         int items[max], top;
         void   create(stack*);
         void   push(stack*, int);
         int    pop(stack*);
};

void   stack::create(stack *p)
{
       p-> top = -1;
}
void    stack::push(stack *p, int ele)
{
       if (p-> top == max-1)
       {
            cout << "\nStack overflows.." << endl;
       }
       else
       {
            p-> items[++p-> top] = ele;
       }
}

int    stack::pop(stack *p)
{
       return  (p-> top == -1)? 0 : p-> items[p-> top--];
}
void  main()
{
       clrscr();
```

```
        char  str[max], rev[max];
        int   i = 0, j = 0;
        stack  obj, q;
        stack  *p = &q;

        obj.create(p);

        cout << "Enter any string : ";
        cin >> str;

        while (str[i] != '\0')
        {
              obj.push(p, str[i]);
              i++;
        }

        while (p-> top != -1)
        {
              rev[j++] = obj.pop(p);
        }
        rev[j] = '\0';

        (strcmp(str, rev) == 0)? cout << "Given string is Palindrome" :
cout << "Given string is not Palindrome";

        getch();
}
```

```
Output -->
Enter any string : eye
Given string is Palindrome

Enter any string : SSP
Given string is not Palindrome
```


4) Write a program to convert decimal number into binary number by using stack.
-->

```
#include<iostream.h>
#include<conio.h>
#define  max  50

class   stack{
        public:
           int items[max], top;
           void  create(stack*);
           void  push(stack*, int);
           int   pop(stack*);
};
void  stack::create(stack *p)
{
        p->top = -1;
}
void  stack::push(stack *p, int ele)
{
        if(p->top == max-1)
        {
```

```
                cout << "Stack overflows.." << endl;
        }
        else
        {
                p->items[++p->top] = ele;
        }
}

int  stack::pop(stack *p)
{
        if(p->top == -1)
        {
                cout << "Stack underflows.." << endl;
                return 0;
        }
        else
        {
                return p->items[p->top--];
        }
}

void  main()
{
    clrscr();

    int num, j = 0, rem, result = 0;
    char binary[max];

    stack  obj, q;
    stack  *p = &q;

    obj.create(p);

    cout << "Enter any number : ";
    cin >> num;

    int num1 = num;

    while (num != 0)
    {
          rem = num % 2;
          obj.push(p, rem);
          num /= 2;
    }

    while(p->top != -1)
    {
         binary[j++] = obj.pop(p) + '0';
    }

    binary[j] = '\0';

    cout << "Binary format of your number "<< num1 << " is : " << binary
<< endl;

    getch();
}

Output -->
Enter any number : 10
```

```
Binary format of your number 10 is : 1010
```

5) Write a program that counts total number of vowels present in string by using stack.

-->

```
#include<iostream.h>
#include<conio.h>
#define   max   50


class  stack
{
       public:
           int items[max], top;
           void  create(stack*);
           void  push(stack*, int);
           int   pop(stack*);
};
void stack::create(stack *p)
{
        p->top = -1;
}
void stack::push(stack *p, int ele)
{
        if(p->top == max-1)
        {
              cout << "Stack overflows.."<< endl;
        }
        else
        {
              p->items[++p->top] = ele;
        }
}
int stack::pop(stack *p)
{
         if(p->top == -1)
         {
             cout << "\nStack underflows.." << endl;
             return 0;
         }
         else
         {
             return p->items[p->top--];
         }
}
void  main()
{
    clrscr();
    char  str[max],ch;
    int i = 0, counter = 0;
    stack  obj, q;
    stack  *p = &q;

    cout << "Enter any string : ";
    cin >> str;

    obj.create(p);
```

```
    while (str[i] != '\0')
    {
          obj.push(p, str[i]);
          i++;
    }


    while (p->top != -1)
    {
        ch = obj.pop(p);
        if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u'
||
            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch ==
'U') {
            counter++;
        }
    }

    cout << "Total number of vowels in the string: " << counter << endl;


     getch();
}
```

```
Output -->
Enter any string : Shreyash
Total number of vowels in the string: 2
```


6) Write a program which converts infix expression into prefix expression.

```
-->

#include<iostream.h>
#include<conio.h>
#include<string.h>
#define  max  50

class  stack
{
    public:
        int items[max], top;
        void  create(stack*);
        void  push(stack*, int);
        int   pop(stack*);
};

void  stack::create(stack *p)
{
      p->top = -1;
}

void  stack::push(stack *p, int ele)
{
      if (p->top == max-1)
      {
           cout << "Stack Overflows.." << endl;
      }
```

```
      else
      {
           p->items[++p->top] = ele;
      }
}

int  stack::pop(stack *p)
{
      if (p->top == -1)
      {
           cout << "Stack underflows.." << endl;
           return 0;
      }
      else
      {
           return p->items[p->top--];
      }
}

void  main()
{

    clrscr();
    char  infix[max], prefix[max], ch;
    int i = 0, j = 0;

    cout << "Enter any infix expression : ";
    cin >> infix;

    stack obj, p, q;
    stack *op_stack = &p;
    stack *oprnd_stack = &q;

    obj.create(op_stack);
    obj.create(oprnd_stack);

    strrev(infix);

    while (infix[i] != '\0')
    {
         if (infix[i] == ')' || infix[i] == '+' || infix[i] == '-' ||
infix[i] == '*' || infix[i] == '/' || infix[i] == '%' || infix[i] == '$'
|| infix[i] == '^')
         {
                 obj.push(op_stack, infix[i]);
         }

         else if (infix[i] == '(')
         {
               while (op_stack->top != -1)
               {
                     ch = obj.pop(op_stack);
                     if (ch != ')')
                     {
                           obj.push(oprnd_stack, ch);
                     }
               }
         }
         else
         {
```

```
                obj.push(oprnd_stack,infix[i]);
          }
          i++;
     }


     while (op_stack->top != -1)
     {
           ch = obj.pop(op_stack);
           if (ch != ')')
           {
                 obj.push(oprnd_stack, ch);
           }
     }

     while (oprnd_stack-> top != -1)
     {
           prefix[j++] = obj.pop(oprnd_stack);
     }

     prefix[j] = '\0';

     cout << "Prefix expression : " << prefix << endl;

     getch();
}

Output -->
Enter any infix expression : ((a*b)^(m-n)+z)/x
Prefix expression : ^*ab/+-mnzx
```


7) Write a program which converts infix expression into postfix expression.

-->

```
#include<iostream.h>
#include<conio.h>
#define  max  50

class  stack
{
     public:
         int items[max], top;
         void  create(stack*);
         void  push(stack*, int);
         int   pop(stack*);
};
void  stack::create(stack *p)
{
      p->top = -1;
}
void  stack::push(stack *p, int ele)
{
      if (p->top == max-1)
      {
           cout << "Stack Overflows.." << endl;
      }
      else
```

```
      {
            p->items[++p->top] = ele;
      }
}
int  stack::pop(stack *p)
{
      if (p->top == -1)
      {
            cout << "Stack underflows.." << endl;
            return 0;
      }
      else
      {
            return p->items[p->top--];
      }
}
void  main()
{
     clrscr();

     char  infix[max], postfix[max], ch;
     int i = 0, j = 0;

     stack obj, q;
     stack *p = &q;
     obj.create(p);

     cout << "Enter any infix string : ";
     cin >> infix;

     while (infix[i] != '\0')
     {
           if (infix[i] == '(' || infix[i] == '+' || infix[i] == '-' ||
infix[i] == '*' || infix[i] == '/' || infix[i] == '%' || infix[i] == '$'
|| infix[i] == '^')
           {
                  obj.push(p, infix[i]);
           }

           else if (infix[i] == ')')
           {
                  while (p->items[p->top] != '(')
                  {
                        ch = obj.pop(p);
                        if (ch != '(') {
                              postfix[j++] = ch;
                        }
                  }
                  obj.pop(p);
           }
           else
           {
                 postfix[j++] = infix[i];
           }
           i++;
     }

     while (p->top != -1)
     {
           ch = obj.pop(p);
```

```
            if (ch != '(')
            {
                 postfix[j++] = ch;
            }
      }

      postfix[j] = '\0';

      cout << "Prefix expression : " << postfix << endl;

      getch();

}

Output -->
Enter any infix string : (x$((a-b)^c)%d)+(m-n)^p
Prefix expression : xab-c^d%$mn-p^+



8) Write a program which check entered expression is valid or not.
-->

#include<iostream.h>
#include<conio.h>
#define  max  50

class  stack
{
    public:
        int items[max], top;
        void  create(stack*);
        void  push(stack*, int);
        int   pop(stack*);
};
void  stack::create(stack *p)
{
      p->top = -1;
}
void  stack::push(stack *p, int ele)
{
      if (p->top == max-1)
      {
           cout << "Stack Overflows.." << endl;
      }
      else
      {
           p->items[++p->top] = ele;
      }
}
int  stack::pop(stack *p)
{
      if (p->top == -1)
      {
           cout << "Stack underflows.." << endl;
           return 0;
      }
      else
      {
           return p->items[p->top--];
```

```
      }
}

void  main()
{
     clrscr();

     char exp[max], ch;

     int i = 0, temp = 0;

     cout << "Enter any exp. which is parentisized : ";
     cin >> exp;

     stack obj, q;
     stack *p = &q;

     obj.create(p);

     while (exp[i] != '\0')
     {
           if (exp[i] == '(' || exp[i] == '[' || exp[i] == '{')
           {
                 obj.push(p, exp[i]);
           }
           else if (exp[i] == ')' || exp[i] == ']' || exp[i] == '}')
           {
                 ch = obj.pop(p);

                 if ((ch == '(') != (exp[i] == ')')
                 || (ch == '[') != (exp[i] == ']')
                 || (ch == '{') != (exp[i] == '}'))
                 {
                       temp = 1;
                       break;
                 }
           }
           i++;
     }

     if (temp == 1 || p->top != -1)
     {
           cout << "Expression is Invalid.." << endl;
     }
     else
     {
           cout << "Expression is Valid.." << endl;
     }
     getch();
}
```

```
Output -->
Enter any exp. which is parentisized : [a(b-c]*d)
Expression is Invalid..

Enter any exp. which is parentisized : [(a-b)*c]
Expression is Valid..
```

9) Write a program that evaluates entered postfix expression.
-->

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
#define  max  50

class  stack
{
    public:
        int items[max], top;
        void  create(stack*);
        void  push(stack*, int);
        int   pop(stack*);
};
void  stack::create(stack *p)
{
      p->top = -1;
}
void  stack::push(stack *p, int ele)
{
      if (p->top == max-1)
      {
           cout << "Stack Overflows.." << endl;
      }
      else
      {
           p->items[++p->top] = ele;
      }
}
int  stack::pop(stack *p)
{
      if (p->top == -1)
      {
           cout << "Stack underflows.." << endl;
           return 0;
      }
      else
      {
           return p->items[p->top--];
      }
}
void  main()
{
     clrscr();

     char postfix[max];
     int i = 0, op1, op2, val;

     cout << "Enter any postfix exp. : ";
     cin >> postfix;

     stack  obj, q;
     stack  *p = &q;
     obj.create(p);

     while (postfix[i] != '\0')
     {
```

```
            if (postfix[i] == '+' || postfix[i] == '-' || postfix[i] ==
'*' || postfix[i] == '/' || postfix[i] == '%' || postfix[i] == '$')
            {
                    op2 = obj.pop(p);
                    op1 = obj.pop(p);

                    switch (postfix[i])
                    {
                            case '+':
                                 val = op1 + op2;
                                 obj.push(p, val);
                                 break;
                            case '-':
                                 val = op1 - op2;
                                 obj.push(p, val);
                                 break;
                            case '*':
                                 val = op1 * op2;
                                 obj.push(p, val);
                                 break;
                            case '/':
                                 val = op1 / op2;
                                 obj.push(p, val);
                                 break;
                            case '%':
                                 val = op1 % op2;
                                 obj.push(p, val);
                                 break;
                            case '$':
                                 val = pow(op1, op2);
                                 obj.push(p, val);
                                 break;
                    }
            }
            else
            {
                  obj.push(p, postfix[i] - 48);
            }
            i++;
     }

     cout << "Value of your postfix exp. ( "<<postfix <<" ) is : " <<
obj.pop(p) << endl;

     getch();
}

Output -->
Enter any postfix exp. : 452*+
Value of your postfix exp. ( 452*+ ) is : 14



10) Write a program to calculate factorial of entered number by using
recursion.
-->

#include<iostream.h>
#include<conio.h>
```

```
int  fact(int n)
{
    return (n == 1 || n == 0)? 1 : n * fact(n-1);
}
void  main()
{
    clrscr();
    int num;
    cout << "Enter any number to find factorial : ";
    cin >> num;

    cout << "Factorial of number " << num << " is : " << fact(num) <<
endl;
    getch();
}

Output -->
Enter any number to find factorial : 5
Factorial of number 5 is : 120
```

11) Write a program to calculate digit sum of entered number by using recursion.
-->

```
#include<iostream.h>
#include<conio.h>

int  digit_sum(int num)
{
        return (num == 0)? 0 : num % 10 + digit_sum(num / 10);
}

void  main()
{
    clrscr();
    int num;
    cout << "Enter any number to find digit sum : ";
    cin >> num;

    cout << "Digit sum of entred no. " << num << " is : " <<
digit_sum(num) << endl;
    getch();
}

Output -->
Enter any number to find digit sum : 123
Digit sum of entred no. 123 is : 6
```

12) Write a program to find face value of entered number by using recursion.
-->

```
#include<iostream.h>
#include<conio.h>

int  face_val(int num)
```

```
{
        if (num == 0)
        {
            return 0;
        }
       face_val(num / 10);
       cout << "Face value of " << num % 10 << " is : " << num % 10 <<
endl;
}
void  main()
{
    int num;
    cout << "Enter any number to find digit sum : ";
    cin >> num;

    face_val(num);
    getch();
}
```

```
Output -->
Enter any number to find digit sum : 123
Face value of 1 is : 1
Face value of 2 is : 2
Face value of 3 is : 3
```

13) Write a program that demonstrate implementation of multiple stacks. (Hint: Take one stack to store even numbers and other stack to store odd numbers.)

```
-->

#include<iostream.h>
#include<conio.h>
#define  max  50

class  stack
{
    public:
        int items[max], top;
        void  create(stack*);
        void  push(stack*, int);
        int   pop(stack*);
};
void  stack::create(stack *p)
{
      p->top = -1;
}
void  stack::push(stack *p, int ele)
{
      if (p->top == max-1)
      {
           cout << "Stack Overflows.." << endl;
      }
      else
      {
           p->items[++p->top] = ele;
      }
}
int  stack::pop(stack *p)
```

```
{
      if (p->top == -1)
      {
           cout << "Stack underflows.." << endl;
           return 0;
      }
      else
      {
           return p->items[p->top--];
      }
}
void  main()
{
    clrscr();
    int  end;
    cout << "Enter end value inbetween you want to find odd or even
numbers : ";
    cin >> end;

    stack obj, p, q;
    stack *even_num = &p;
    stack *odd_num = &q;

    obj.create(even_num);
    obj.create(odd_num);

    while (end != 1)
    {
          if (end % 2 == 0)
          {
               obj.push(even_num, end);
          }
          else
          {
               obj.push(odd_num, end);
          }
          end--;
    }

    cout << "\nEven numbers : ";
    while (even_num->top != -1)
    {
        cout << obj.pop(even_num) << "   ";

    }

    cout << "\nOdd numbres : ";
    while (odd_num->top != -1)
    {
        cout << obj.pop(odd_num) << "   ";
    }

    getch();
}

Output -->
Enter end value inbetween you want to find odd or even numbers : 10

Even numbers : 2   4   6   8   10
Odd numbres : 3   5   7   9
```

Created by SPatil..