**1)(a)**

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<string.h>

class stack{

    public:

     char exp[20], infix[30], prefix[30];

     int top;

     stack()

     {top = -1;}

     int isoperator(char);

     void push(char s)

     { if(top==20)

       {cout<<"overflow"<<endl;

return;

       }

       exp[++top] = s;

     }

     char pop()

     { if(top==-1)

       {

return '\_';

       }

       return (exp[top--]);

     }

     void read()

     { cout<<"Enter the expression"<<endl;

       cin>>infix;

     }

     void conv();

     int prec(char s);

};

int stack::prec(char s)

{             switch(s)

      { case '%' : return 1;

       case '$' : return 2

       case '#' : return 3;

               case'@': return 4;

       case '(' : return 0;

       default  : return -1;

      }

}

int stack::isoperator(char s)

{             switch(s)

      {case '%' :

       case '$' :

       case '#' :

       case '@' :

       case '(' : return 1;

       default  : return 0;

      }

}

void stack :: conv()

{

int i,sym,j=0;

strrev(infix);

exp[++top]='#';

for(i=0;i<strlen(infix);i++)

{

sym=infix[i];

if(isoperator(sym)==0)

{

prefix[j]=sym;

j++;

}

else

{

if(sym==')')

push(sym);

else if(sym=='(')

{

while(exp[top]!=')')

{

prefix[j]=pop();

j++;

}

pop();

}

else

{

if(prec(sym)>prec(exp[top]))

push(sym);

else

{

while(prec(sym)<=prec(exp[top]))

{

prefix[j]=pop();

j++;

}

push(sym);

}

}

}

}

while(exp[top]!='\_')

{

prefix[j]=pop();

j++;

}

prefix[j]='\0';

strrev(prefix);

cout<<"Prefix :"<<prefix;

getch();

}

void main()

{

stack obj;

clrscr();

obj.read();

obj.conv();

getch();

}

**(b)**

Ex: a@b#(c%d)

Reversed infix expression:  )d%c(#b@a

Steps               current symbol                      str                       stack

1.                                            )                                                              )

2.                                            d                             d                             )

3.                                            %                            d                             )%

4.                                            c                              dc                           )%

5.                                            (                              dc%                        )

6.                                            #                             dc%                        )#

7.                                            b                             dc%b                     )#

8.                                            @                            dc%b                     )#@

9.                                                   NULL                     dc%b@#

Reverse the output

Answer= #@b%cd

**2)(a)**

#include<iostream.h>

#include<conio.h>

#include<ctype.h>

class stack{

    public:

     char exp[20], postfix[30];

     int top;

     stack()

     {top = -1;}

     void push(char s)

     {

       exp[++top] = s;

     }

     char pop()

     { if(top==-1)

       {

return '#';

       }

       return (exp[top--]);

     }

     void read()

     { cout<<"enter expression"<<endl;

       cin>>postfix;

     }

     int conv();

     int op(int, int, char);

   };

int stack::op(int a, int b, char opr)

{             switch(opr)

      {case '+' : return a+b;

       case '-' : return a-b;

       case '\*' : return a\*b;

       case '/' : return a/b;

       default  : return 0;

      }

}

int stack::conv()

{              int op1, op2, i=0;

       while(postfix[i]!='\0')

       {if(isdigit(postfix[i]))

  push(postfix[i] - '0');

else

{ op2 = pop();

  op1 = pop();

  int res = op(op1, op2, postfix[i]);

  push(res);

}

i++;

       }

       return pop();

}

void main()

{stack obj; int a;

 clrscr();

 obj.read();

 a=obj.conv();

 cout<<"Answer = "<<a<<endl;

 getch();

}

**(2)(b)**

Evaluating postfix expression using stack

Steps:

1.Scan the postfix expression from left to right

2.If an operand is encountered, push it on to the stack.

3.If an operator is encountered, pop the top two elements from the stack, perform the required operation based on operator and push the result back on stack.

4.Perform this operation until end of expression is reached.

5.At the end, stack contains the final result.

Ex:-623+-

Symbol op1 op2 op1 operator op2 stack

6 6

2 6,2

3 6,2,3

+ 2 3 5 6,5

- 6 5 1 1

Ans=1

**3)(i&ii)**

#include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<string.h>

class node{ int marks;

char name[20];

int reg\_no;

node \*next;

node \*head;

node \*tail;

public:

node()

{ head = NULL;

tail = NULL;

}

void insert();

void disp();

void sort();

};

void node::insert()

{node \*temp = new node;

temp->next = NULL;

cout<<"enter name : ";

gets(temp->name);

cout<<"enter registration no. : ";

cin>>temp->reg\_no;

cout<<"enter marks :";

cin>>temp->marks;

if(head!=NULL && tail!=NULL)

{tail->next = temp;

tail = temp;

}

else

tail=head=temp;

}

void node::disp()

{node \*temp = head;

if(temp==NULL)

cout<<"empty!"<<endl;

while(temp!=NULL)

{cout<<"name : "<<temp->name<<endl;

cout<<"registration no : "<<temp->reg\_no<<endl;

cout<<"marks : "<<temp->marks<<endl;

temp=temp->next;

}

}

void node::sort()

{int s;

do

{ int t; char n[20];

node \*temp = head;

s=0;

while(temp!=NULL)

{ if(temp->reg\_no > temp->next->reg\_no)

{ t = temp->reg\_no;

temp->reg\_no = temp->next->reg\_no;

temp->next->reg\_no = t;

int m = temp->marks;

temp->marks = temp->next->marks;

temp->next->marks = m;

strcpy(n,temp->name);

strcpy(temp->name,temp->next->name);

strcpy(temp->next->name,n);

s=1;

}

temp = temp->next;

}

}while(s);

}

void main()

{node obj;

int n;

clrscr();

cout<<"no. of students : ";

cin>>n;

for(int i=0 ; i<n ; i++)

obj.insert();

obj.disp();

obj.sort();

cout<<"After Sorting :-"<<endl;

obj.disp();

getch();

}

**4)(a & b)**

#include<iostream.h>

#include<conio.h>

#include<process.h>

class node{ int data;

node \*next;

node \*head, \*tail;

public:

node()

{head = NULL;

tail = NULL;

}

void insert();

void disp();

void del\_odd();

void rev();

};

void node::insert()

{node \*temp = new node;

temp->next = NULL;

cout<<"enter value"<<endl;

cin>>temp->data;

if(head!=NULL && tail!=NULL)

{tail->next=temp;

tail=temp;

}

else

tail=head=temp;

}

void node::disp()

{node \*temp=head;

if(temp==NULL)

cout<<"empty!"<<endl;

while(temp!=NULL)

{cout<<temp->data<<"\t";

temp=temp->next;

}

}

void node::del\_odd()

{node \*temp=head, \*tempr;

tempr=NULL;

while(head->data%2!=0)

{

head=head->next;

temp=temp->next;

}

while(temp!=NULL)

{if(temp->data%2!=0)

{node \*d=temp;

tempr->next=temp->next;

temp=temp->next;

delete(d);

}

else

{tempr=temp;

temp=temp->next;

}

}

}

void node::rev()

{node \*temp=head,\*t, \*r, \*tempr;

int count=0 , a=0 , c=1;

tempr = NULL;

while(temp!=NULL)

{temp=temp->next;

count++;

}

tempr =NULL;

a=count/2;

temp=head;

while(c<=a)

{tempr=temp;

temp=temp->next;

c++;

}

t=tempr;

while(temp!=NULL)

{r=temp->next;

temp->next=tempr;

tempr=temp;

temp=r;

}

t->next=tempr;

while(tempr->next!=t)

{tempr=tempr->next;}

tempr->next=NULL;

}

void main()

{node obj;

int ch,n;

clrscr();

cout<<"enter no. of elements:-"<<endl;

cin>>n;

for(int i=0 ; i<n ; i++)

obj.insert();

obj.disp();

cout<<endl;

//obj.rev();

//obj.disp();

obj.del\_odd();

obj.disp();

getch();

}

**5(a)**

Circular queue is a bounded queue which implements arrays.  
  
It is beter than a normal queue because in this we can effectively utilise the memory space.If we have a normal queue and have deleted some elements from there then empty space is created there and even if the queue has empty cells then also we cannot insert any new element because the insertion has to be done from one side only(i.e rear or tail) and deletion has to be done from another side(i.e front or head).But in case of circular queue the front and rear are adjacent to each other.

**5(b)**

**#include<iostream.h>**

**#include<conio.h>**

**#include<process.h>**

**class queue**

**{**

**int front,rear;**

**int q[20];**

**public:**

**void insertq(int);**

**int delq();**

**void display();**

**queue()**

**{**

**front=0;**

**rear=-1;**

**}**

**};**

**void queue::insertq(int n)**

**{**

**if(rear==n-1)**

**{**

**cout<<"Queue overflow \n";**

**return;**

**}**

**int item==0;**

**cout<<"Enter item to be inserted\n";**

**cin>>item;**

**rear=rear+1;**

**q[rear]=item;**

**}**

**int queue::delq()**

**{**

**int io=0;**

**if(front>rear)**

**return -1;**

**cout<<"Which end do you want to delete.Enter 1 for front and 2 for rear\n";**

**cin>>io;**

**if(io==1)**

**return q[front++];**

**else**

**return q[rear--];**

**}**

**void queue::display()**

**{**

**int i;**

**if(front>rear)**

**{**

**cout<<"Empty queue\n";**

**return;**

**}**

**cout<<"Contents:";**

**for(i=front;i<=rear;i++)**

**cout<<q[i];**

**}**

**void main()**

**{**

**int a,ch,size;**

**queue obj;**

**clrscr();**

**cout<<"Enter size:\n";**

**cin>>size;**

**while(1)**

**{**

**cout<<"1.Enter\n2.Delete\n3.Display\n4.Exit\n";**

**cout<<"Enter your choice";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1: obj.insert(size);break;**

**case 2: a=obj.delete();**

**if(a==-1)**

**cout<<"List empty";**

**else**

**cout<<"Item deleted is "<<a;**

**break;**

**case 3: obj.display();**

**break;**

**case 4: exit(0);**

**default: cout<<"Wrong choice";**

**}**

**getch();**

**}**

**}**