```
#include <iostream>

using namespace std;

const int size = 10;

void push(int stack[size], int &top)
{
    int value;
    if(top == size-1)
        cout<<"stack is full, insertion is not possible\n";

    else
    {
        cout<<"Enter any number : ";
        cin>>value;

        ++top;
        stack[top] = value;
    }
}

void pop(int stack[size], int &top)
{
    int value;
    if(top == -1)
        cout<<"stack is empty, deletion is not possible\n";
    else
    {
        value = stack[top];
        --top;
    }
}

void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

void con_array_even(int stack[size], int a[10], int top)
{
    int i, j=0;
    cout<<"Even values from the stack : \n";
    for(i=0;i<=top;i++)
    {
        if(stack[i]%2==0)
        {
            a[j]=stack[i];
            cout<<a[j]<<"   ";
            ++j;
        }
    }
    cout<<"\n";
}

void con_array_odd(int stack[size], int b[10], int top)
{
    int i, j=0;
    cout<<"Odd values from the stack : \n";
    for(i=0;i<=top;i++)
    {
        if(stack[i]%2!=0)
        {
            b[j]=stack[i];
            cout<<b[j]<<"   ";
            ++j;
        }
    }
    cout<<"\n";
}


int main()
{
    int stack[size];
    int top = -1;
    int x;
    int a[10],b[10];

    for( ; x != 6; )
      {
            cout<<"1- push \n";
            cout<<"2- pop \n";
            cout<<"3- print \n";
            cout<<"4-convert even values from stack to one dimensional array\n";
        cout<<"5-convert odd values from stack to  one dimensional array \n";
        cout<<"6-Exit\n";

            cout<<"Enter your choice : ";
            cin>>x;

            switch (x)
                  {
                        case 1:push(stack, top); break;
                        case 2:pop(stack, top); break;
                        case 3:print(stack, top); break;
                        case 4:con_array_even(stack, a, top);break;
                        case 5:con_array_odd(stack, b, top);break;
                        default:cout<<"Error\n";
                  }

      }

    return 0;
}
```

```
#include <iostream>

using namespace std;

const int size = 6;

void push(int stack[size], int &top)
{
    int value;
    if(top == size-1)
        cout<<"stack is full, insertion is not possible\n";

    else
    {
        cout<<"Enter any number : ";
        cin>>value;

        ++top;
        stack[top] = value;
    }
}

void pop(int stack[size], int &top)
{
    int value;
    if(top == -1)
        cout<<"stack is empty, deletion is not possible\n";
    else
    {
        value = stack[top];
        --top;
    }
}

void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

void prime(int stack[size], int &top)
{
      int i, j, x;

      for(i = 0; i <= top; i++)
      {
            if(stack[i] != 1)
            {
                  x = 0;
                  for(j =2; j < stack[i]; j++)
                  {
                        if(stack[i] % j == 0)
                              x = 1;
                  }

                  if(x == 0)
                        cout<<stack[i]<<"\t";
            }

            else
                  cout<<stack[i]<<"\t";
      }
      cout<<"\n";
}

int main()
{
    int stack[size];
    int top = -1;
    int x;

    for( ; x != 5; )
      {
            cout<<"1- push \n";
            cout<<"2- pop \n";
            cout<<"3- print \n";
            cout<<"4- prime \n";
            cout<<"5- exit \n";

            cout<<"Enter your choice : ";
            cin>>x;

            switch (x)
                  {
                        case 1:push(stack, top); break;
                        case 2:pop(stack, top); break;
                        case 3:print(stack, top); break;
                        case 4:prime(stack, top); break;
                        default:cout<<"Error\n";
                  }

      }

    return 0;
}
```

```
#include <iostream>

using namespace std;

const int size = 7;

void push(int stack[size], int &top)
{
    int value;
    int i;

    for(i=0; i<=5; i++)
            {
                  cout<<"Enter any number : ";
                  cin>>value;

                  ++top;
                  stack[top]=value;
            }
}

void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

int main()
{
    int stack[size];
    int top = -1;

    push(stack, top);
    print(stack, top);


    return 0;
}
```

لديك stack المطلوب تحويل القيم الى stack 10 حجمه 5 ثاني يحتوي على 3 قيم علما ان حجمه .

Q/You have a stack of size 5 to convert the values into a second stack of 3 values, with a size of 10.

-

```
#include <iostream>

using namespace std;

const int size1 = 5;
const int size2 = 10;

void push(int stack[size1], int &top)
{
    int value;
    if(top == size1-1)
        cout<<"stack is full, insertion is not possible\n";

    else
    {
        cout<<"Enter any number : ";
        cin>>value;

        ++top;
        stack[top] = value;
    }
}

void print(int stack[size1], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }

}

void pushst2(int stack[size1], int stack2[size2], int &top, int &top2)
{
      int i;

      for(i=0; i<=top; i++)
      {
            ++top2;
            stack2[top2] = stack[i];
      }
}

void print2(int stack2[size2], int top2)
```

```
{
    int i;
    if(top2 == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top2; i++)
        {
            cout<<stack2[i]<<"\t";
        }
        cout<<"\n";
    }

}

int main()
{
    int stack[size1], stack2[size2]={1,2,3};
    int top = -1;
    int top2 = 2;
    int x;

    for( ; x != 5; )
      {
            cout<<"1- push \n";
            cout<<"2- print \n";
            cout<<"3-convert to stack 2 \n";
            cout<<"4-print stack 2 \n";
        cout<<"5-Exit\n";

            cout<<"Enter your choice : ";
            cin>>x;

            switch (x)
                  {
                        case 1:push(stack, top); break;
                        case 2:print(stack, top); break;
                        case 3:pushst2(stack, stack2, top, top2);break;
                        case 4:print2(stack2, top2);break;
                        default:cout<<"Error\n";
                  }

      }

    return 0;
}
```

```
#include <iostream>

using namespace std;

const int size = 8;

void pop(int stack[size], int &top)
{
      int value;
      int i;

      for(i=0; i<=3; i++)
      {
            value=stack[top];
            --top;
      }

}

void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

int main()
{
    int stack[size]={1,2,3,4,5,6};
    int top = 5;

    print(stack, top);
    cout<<"\n";
    pop(stack, top);
    cout<<"\n";
    print(stack, top);


    return 0;
}
```

```
#include <iostream>

using namespace std;

const int size = 8;

void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

int main()
{
    int stack[size]={3,5,8,4,9,7};
    int top = 5;

    print(stack, top);


    return 0;
}
```

Q2:) You have stack of size (10) contain (5) element , Write program segment with draw to:
a) Add (7) elements to stack ?
b) convert even values to another empty stack of size (6)?
c) print the final state for new stack ?

```
#include <iostream>

using namespace std;

const int size = 10;

void push(int stack[size], int &top)
{
    int value, x, i;

    for(i=0; i<=6; i++)
      {
            if(top == size-1)
            {
                  x=stack[top];
                  --top;
                  cout<<"Enter any number : ";
                  cin>>value;

                  ++top;
                  stack[top] = value;
            }

            else
            {
                  cout<<"Enter any number : ";
                  cin>>value;

                  ++top;
                  stack[top] = value;
            }
      }
}


void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

void con_stack_even(int stack[size], int a[10], int top)
```

```
{
    int i, j=0;
    cout<<"Even values from the stack : \n";
    for(i=0;i<=top;i++)
    {
        if(stack[i]%2==0)
        {
            a[j]=stack[i];
            cout<<a[j]<<"   ";
            ++j;
        }
    }
    cout<<"\n";
}


int main()
{
    int stack[size]={1,2,3,4,5};
    int top = 4;
    int a[6];

    push(stack, top);
    print(stack, top);
    con_stack_even(stack, a, top);

    return 0;
}
```

write program to split the content of stack S into two stacks one for numbers larger than 50 and the other for numbers smaller or equal to 50.

```
#include <iostream>

using namespace std;

const int size = 10;

void push(int stack[size], int &top)
{
    int value;
    if(top == size-1)
        cout<<"stack is full, insertion is not possible\n";

    else
    {
        cout<<"Enter any number : ";
        cin>>value;

        ++top;
        stack[top] = value;
    }
}

void pop(int stack[size], int &top)
{
    int value;
    if(top == -1)
        cout<<"stack is empty, deletion is not possible\n";
    else
    {
        value = stack[top];
        --top;
    }
}

void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

void larg(int stack[size], int top, int larager[size])
{
      int i, j=0;

      for(i=0; i<=top; i++)
      {
            if(stack[i] > 50)
            {
                  larager[j] = stack[i];
                  cout<<larager[j]<<"\t";
                  ++j;
            }
      }
      cout<<"\n";
}

void small(int stack[size], int top, int smaller[size])
{
      int i, j=0;

      for(i=0; i<=top; i++)
      {
            if(stack[i] <= 50)
            {
                  smaller[j] = stack[i];
                  cout<<smaller[j]<<"\t";
                  ++j;
            }
      }
      cout<<"\n";
}


int main()
{
    int stack[size],larager[size], smaller[size];
    int top = -1;
    int x;

    for( ; x != 6; )
      {
            cout<<"1- push \n";
            cout<<"2- pop \n";
            cout<<"3- print \n";
            cout<<"4-stack for numbers larger than 50 \n";
            cout<<"5-stack for numbers smaller or equal to 50 \n";
        cout<<"6-Exit\n";

            cout<<"Enter your choice : ";
            cin>>x;

            switch (x)
                  {
                        case 1:push(stack, top); break;
                        case 2:pop(stack, top); break;
                        case 3:print(stack, top); break;
                        case 4:larg(stack, top, larager);break;
                        case 5:small(stack, top, smaller);break;
                        default:cout<<"Error\n";
                  }

      }

    return 0;
```

```
}
```

```
#include <iostream>

using namespace std;

const int size = 6;

void push(int stack[size], int &top)
{
    int value;
    if(top == size-1)
        cout<<"stack is full, insertion is not possible\n";

    else
    {
        cout<<"Enter any number : ";
        cin>>value;

        ++top;
        stack[top] = value;
    }
}

void pop(int stack[size], int &top)
{
    int value;
    if(top == -1)
        cout<<"stack is empty, deletion is not possible\n";
    else
    {
        value = stack[top];
        --top;
    }
}

void print(int stack[size], int top)
{
    int i;
    if(top == -1)
        cout<<"stack is empty, printing is not possible\n";
    else
    {
        for(i=0; i<=top; i++)
        {
            cout<<stack[i]<<"\t";
        }
        cout<<"\n";
    }
}

int main()
{
    int stack[size];
    int top = -1;
    int x;

    for( ; x != 4; )
      {
            cout<<"1- push \n";
            cout<<"2- pop \n";
            cout<<"3- print \n";
            cout<<"4- exit \n";

            cout<<"Enter your choice : ";
            cin>>x;

            switch (x)
                  {
                        case 1:push(stack, top); break;
                        case 2:pop(stack, top); break;
                        case 3:print(stack, top); break;
                        default:cout<<"Error\n";
                  }

      }

    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

const int size1 = 8;
const int size2 = 5;

void con_stack(int stack[size2], int &top, int q[size1], int &f, int &r)
{
      int i, value;

      for(i = 0; i <= r; i++)
      {
            if(q[i] > 100)
            {
                  if(top == size1 - 1)
                  {
                        value = stack[top];
                        --top;

                        ++top;
                        stack[top] = q[i];
                  }

                  else
                  {
                        ++top;
                        stack[top] = q[i];
                  }
            }
      }
}

void prints(int stack[size2], int top)
{
      int i;
      cout<<"stack larger 100 : \n";
      for(i = 0; i <= top; i++)
            cout<<stack[i]<<"\t";
      cout<<"\n";
}

void printq(int q[size1], int f, int r)
{
      int i;
      cout<<"Queue elements : \n";

      if(f == -1)
            cout<<"Queue is empty nothing to print !!";

      else
      {
            for(i = r; i >= f; i--)
            {
                  cout<<q[i]<<"\t";
            }
      }
}
```

```
int main()
{
      int q[size1]={100, 203, 99, 409, 523, 611, 79};
      int stack[size2];
      int top = -1;
      int f = 0;
      int r = 6;

      printq(q, f, r);
      cout<<"\n";

      con_stack(stack, top, q, f, r);
      cout<<"\n";

      prints(stack, top);
      cout<<"\n";


    return 0;
}
```

```
#include <iostream>

using namespace std;

const int size = 6;

void insertion(int q[size], int &f, int &r)
{
      int value;
      if(r == size - 1)
            cout<<"Queue is full !! Insertion is not possible.\n";

      else
      {
            cout<<"Enter any number : ";
            cin>>value;

            ++r;
            q[r] = value;
      }

      if(f == -1)
            f = 0;
}

void deletion(int q[size], int &f, int &r)
{
      int value;
      if(f == -1)
            cout<<"Under flow !! Queue is empty deletion is not possible.\n";

      else if(f == r)
      {
            value = q[f];

            f = -1;
            r = -1;
      }

      else
      {
            value = q[f];
            ++f;
      }
}

void print(int q[size], int f, int r)
{
      int i;

      if(f == -1)
            cout<<"Queue is empty nothing to print !!";

      else
      {
            for(i = r; i >= f; i--)
            {
                  cout<<q[i]<<"\t";
            }
      }
}


int main()
{
      int q[size];
      int f = -1;
      int r = -1;
      int x;

      for( ; x != 4; )
      {
            cout<<"1- insertion\n";
            cout<<"2- deletion\n";
            cout<<"3- print\n";
            cout<<"4- exit\n";

            cout<<"Enter your choice : ";
            cin>>x;
            cout<<"\n";

            switch (x)
            {
                  case 1 :insertion(q, f, r); break;
                  case 2 :deletion(q, f, r); break;
                  case 3 :print(q, f, r); break;
                  default:cout<<"Error\n";
            }
            cout<<"\n";
      }


    return 0;
}
```

Q1/ Having a Queue of size (10), write complete program to split the even values of the queue into other queue of size (5) and the odd values into stack of size (5).

```
#include <iostream>

using namespace std;

int const sizeq1 = 10;
int const sizeq2 = 5;
int const sizest = 5;


void insertion(int q1[sizeq1], int &f1, int &r1)
{
      int value;
      int i;

      for(i = 0; i <= 9; i++)
      {
            if(r1 == sizeq1-1)
            cout<<"Queue is empty, insertion is not possible\n";

            else
            {
                  cout<<"Enter any number for queue : ";
                  cin>>value;

                  ++r1;
                  q1[r1] = value;
            }

            if(f1 == -1)
                  f1 = 0;
            }
}

void spillt_qu_st(int q1[sizeq1], int q2[sizeq2], int stack[sizest], int &top, int
&f1, int &r1, int &f2, int &r2)
{
      int i;

      for(i = r1; i >= f1; i--)
      {
            if(q1[i] % 2 == 0)
            {
                  r2++;
                  q2[r2] = q1[i];

                  if(f2 == -1)
                        f2 = 0;
            }

            else if(q1[i] % 2 != 0)
            {
                  top++;
                  stack[top] = q1[i];
            }
      }
}

void printq1(int q1[sizeq1], int f1, int r1)
{
      int value;
      int i;

      cout<<"\n Queue 1 full : \n";

      if(r1 == -1)
            cout<<"Queue 1 is empty\n";
      else
      {
            for(i = r1; i >= f1; i--)
                  cout<<q1[i]<<"\t";
      }
}

void printq2(int q2[sizeq2], int f2, int r2)
{
      int value;
      int i;

      cout<<"\n Queue 2 even : \n";

      if(r2 == -1)
            cout<<"Queue 2 is empty\n";
      else
      {
            for(i = r2; i >= f2; i--)
                  cout<<q2[i]<<"\t";
      }
}

void printst(int stack[sizest], int top)
{
      int value;
      int i;

      cout<<"\n stack odd : \n";
      if(top == -1)
            cout<<"stack 1 is empty\n";
      else
      {
            for(i = 0; i <= top; i++)
                  cout<<stack[i]<<"\t";
      }
}



int main()
{
      int q1[sizeq1];
      int q2[sizeq2];
      int stack[sizest];
```

```
        int top = -1;

        int f1 = -1;
        int r1 = -1;

        int f2 = -1;
        int r2 = -1;


        insertion(q1, f1, r1);
        cout<<"\n";

        spillt_qu_st(q1, q2, stack, top, f1, r1, f2, r2);
        cout<<"\n";

        printq1(q1, f1, r1);
        cout<<"\n";

        printq2(q2, f2, r2);
        cout<<"\n";

        printst(stack, top);
        cout<<"\n";

    return 0;
}
```

Q5/ Given Queue of size (5) with (5) elements, find the factorial of each value of this queue and put it in an empty stack of size (5) ?

```
#include <iostream>

using namespace std;

int const size = 5;


void insertion(int q[size], int &f, int &r)
{
      int value;
      int i;

      for(i = 0; i <= 4; i++)
      {
            if(r == size-1)
            cout<<"Queue is empty, insertion is not possible\n";

            else
            {
                  cout<<"Enter any number for queue : ";
                  cin>>value;

                  ++r;
                  q[r] = value;
            }

            if(f == -1)
                  f = 0;
            }
}


void print(int q[size], int f, int r)
{
      int value;
      int i;

      if(r == -1)
            cout<<"Queue  is empty\n";
      else
      {
            for(i = r; i >= f; i--)
                  cout<<q[i]<<"\t";
      }
}

void qu_st_fc(int q[size], int stack[size], int &top, int &f, int &r)
{
      int i, j, x;
      int fact = 1;

      for(i = r; i >= f; i--)
      {
            fact = 1;
            x = q[i];
            for(j = 1; j <= x; j++)
            {
                  fact = fact * j;
            }

            ++top;
            stack[top] = fact;

      }
}

void printst(int stack[size], int top)
{
      int i;

      if(top == -1)
            cout<<"Stack is empty !!\n";
      else
      {
            for(i = 0; i <= top; i++)
                  cout<<stack[i]<<"\t";
      }

}


int main()
{
      int q[size];
      int stack[size];

      int top = -1;
      int f = -1;
      int r = -1;

      insertion(q, f, r);
      cout<<"\n Queue\n";

      print(q, f, r);
      cout<<"\n stack\n";

      qu_st_fc(q, stack, top, f, r);
      printst(stack, top);


    return 0;
}
```

Q2/B/ Having a Queue of size (10), write complete program to split the even values of the queue into other queue of size (5) and the odd values into Queue of size (5).

```
#include <iostream>

using namespace std;

int const sizeq1 = 10;
int const sizeq2 = 5;
int const sizeq3 = 5;


void insertion(int q1[sizeq1], int &f1, int &r1)
{
      int value;
      int i;

      for(i = 0; i <= 9; i++)
      {
            if(r1 == sizeq1-1)
            cout<<"Queue is empty, insertion is not possible\n";

            else
            {
                  cout<<"Enter any number for queue : ";
                  cin>>value;

                  ++r1;
                  q1[r1] = value;
            }

            if(f1 == -1)
                  f1 = 0;
            }
}

void spillt_qu_st(int q1[sizeq1], int q2[sizeq2], int q3 [sizeq3], int &f1, int
&r1, int &f2, int &r2, int &f3, int &r3)
{
      int i;

      for(i = r1; i >= f1; i--)
      {
            if(q1[i] % 2 == 0)
            {
                  r2++;
                  q2[r2] = q1[i];

                  if(f2 == -1)
                        f2 = 0;
            }

            else if(q1[i] % 2 != 0)
            {
                  r3++;
                  q3 [r3] = q1[i];
                  if(f3 == -1)
                        f3 = 0;
            }
      }
}

void printq1(int q1[sizeq1], int f1, int r1)
{
      int value;
      int i;

      cout<<"\n Queue 1 full : \n";

      if(r1 == -1)
            cout<<"Queue 1 is empty\n";
      else
      {
            for(i = r1; i >= f1; i--)
                  cout<<q1[i]<<"\t";
      }
}

void printq2(int q2[sizeq2], int f2, int r2)
{
      int value;
      int i;

      cout<<"\n Queue 2 even : \n";

      if(r2 == -1)
            cout<<"Queue 2 is empty\n";
      else
      {
            for(i = r2; i >= f2; i--)
                  cout<<q2[i]<<"\t";
      }
}

void printst(int q3[sizeq3], int f3, int r3)
{
      int value;
      int i;

      cout<<"\n Queue 3 odd : \n";
      if(r3 == -1)
            cout<<"Queue 3 is empty\n";
      else
      {
            for(i = r3; i >= f3; i--)
                  cout<<q3 [i]<<"\t";
      }
}



int main()
{
      int q1[sizeq1];
      int q2[sizeq2];
      int q3[sizeq3];

      int f1 = -1;
      int r1 = -1;

      int f2 = -1;
      int r2 = -1;

      int f3 = -1;
      int r3 = -1;

      insertion(q1, f1, r1);
      cout<<"\n";

      spillt_qu_st(q1, q2, q3 ,f1, r1, f2, r2, f3, r3);
      cout<<"\n";

      printq1(q1, f1, r1);
      cout<<"\n";

      printq2(q2, f2, r2);
      cout<<"\n";

      printst(q3 , f3, r3);
      cout<<"\n";

    return 0;
}
```

```
#include <iostream>

using namespace std;

int const size = 10;

void insertion(int q[size], int &f, int &r)
{
      int value;
      int i;

      for(i = 0; i <= 9; i++)
      {
            if(r == size-1)
            cout<<"Queue is empty, insertion is not possible\n";

            else
            {
                  cout<<"Enter any number for queue : ";
                  cin>>value;

                  ++r;
                  q[r] = value;
            }

            if(f == -1)
                  f = 0;
            }
}

void printq(int q[size], int f, int r)
{
      int i;

      cout<<"Queue  is :\n";
      if(r == -1)
            cout<<"Queue  is empty\n";
      else
      {
            for(i = r; i >= f; i--)
                  cout<<q[i]<<"\t";
      }
}

void con_qu_st_arr(int q[size], int stack[size], int array_even[size], int &top,
int &f, int &r, int &x)
{
      int i, j = 0;

      for(i = r; i >= f; i--)
      {
            if(q[i] %2 == 0)
            {
                  array_even[j] = q[i];
                  ++j;
                  ++x;
            }
            else
            {
                  ++top;
                  stack[top] = q[i];
            }
      }
}

void printst(int stack[size], int top)
{
      int i;
      cout<<"stack ood  is :\n";

      if(top == -1)
            cout<<"Stack is empty \n";

      else
      {
            for(i = 0; i <= top; i++)
                  cout<<stack[i]<<"\t";
      }
}

void printarr(int array_even[size], int x)
{
      int i;
      cout<<"array even  is :\n";

      for(i = 0; i < x; i++)
            cout<<array_even[i]<<"\t";

}

int main()
{
      int q[size];
      int array_even[size];
      int stack[size];

      int top = -1;
      int f = -1;
      int r = -1;
      int x = 0;

      insertion(q, f, r);
      cout<<"\n";

      printq(q, f, r);
      cout<<"\n";

      con_qu_st_arr(q, stack, array_even, top, f, r, x);
      cout<<"\n";

      printarr(array_even, x);
      cout<<"\n";

      printst(stack, top);

    return 0;
}
```

```
#include <iostream>

using namespace std;

const int size = 8;

void insertion(int q[size], int &f, int &r)
{
      int value;
      int i;

      for(i = 0; i<=5; i++)
      {
            if(r == size - 1)
            cout<<"Queue is full !! Insertion is not possible.\n";

            else
            {
                  cout<<"Enter any number : ";
                  cin>>value;

                  ++r;
                  q[r] = value;
            }

            if(f == -1)
                  f = 0;
      }

}

void deletion(int q[size], int &f, int &r)
{
      int value;
      int i;

      for(i = 0; i <= 2; i++)
      {
            if(f == -1)
            cout<<"Under flow !! Queue is empty deletion is not possible.\n";

            else if(f == r)
            {
                  value = q[f];

                  f = -1;
                  r = -1;
            }

            else
            {
                  value = q[f];
                  ++f;
            }
      }
}

void print(int q[size], int f, int r)
{
      int i;

      if(f == -1)
            cout<<"Queue is empty nothing to print !!";

      else
      {
            for(i = r; i >= f; i--)
            {
                  cout<<q[i]<<"\t";
            }
      }
}


int main()
{
      int q[size];
      int f = -1;
      int r = -1;

      insertion(q, f, r);
      cout<<"\n";
      print(q, f, r);
      cout<<"\n";
      deletion(q, f, r);
      cout<<"\n";
      print(q, f, r);
      cout<<"\n";


    return 0;
}
```

Q4/ Given a circlular queue of size (10) contains (10) elements,
write coplete program to convert the even values to any array of size(10).

```
#include <iostream>

using namespace std;

int const size = 10;
int j = 0;

void insertion(int cq[size], int &f, int &r)
{
      int value;
      int i;

      for(i = 0; i <= 9; i++)
      {
            if(r == size - 1 && f == 0)
                  cout<<"C Queue full !! insertion is not possible\n";

            else if(f - r == 1)
                  cout<<"C Queue full !! insertion is not possible\n";

            else if(r == size - 1 && f > 0)
            {
                  cout<<"Enter any value : ";
                  cin>>value;

                  r = 0;
                  cq[r] = value;
            }

            else
            {
                  cout<<"Enter any value : ";
                  cin>>value;

                  ++r;
                  cq[r] = value;
            }

            if(f == -1)
                  f = 0;
      }
}


int con_Cq_arr(int cq[size], int arr_even[size], int &f, int &r)
{
      int value, i;

      if(f == -1)
            cout<<"C Queue is empty\n";

      else if(f == r)
            {
                  if(cq[i] % 2 == 0)
```

```
                {
                        value = cq[r];
                        f = -1;
                        r = -1;

                        arr_even[j] = value;
                        ++j;
                }
        }

    else if(r > f)
    {
        for(i = f; i <= r; i++)
        {
            if(cq[i] % 2 == 0)
            {
                value = cq[i];

                arr_even[j] = value;
                ++j;
            }

        }
    }

    else
    {
        for(i = f; i <= size - 1; i++)
        {
            if(cq[i] % 2 == 0)
            {
                value = cq[i];

                arr_even[j] = value;
                ++j;
            }
        }
        for(i = 0; i <= r; i++)
        {
            if(cq[i] % 2 == 0)
            {
                value = cq[i];

                arr_even[j] = value;
                ++j;
            }
        }
    }

    for(i = 0; i < j; i++)
        cout<<arr_even[i]<<"\t";
    cout<<"\n";
}


void print(int cq[size], int &f, int &r)
{
    int i;
```

```
	if(f == -1)
		cout<<"C Queue is empty\n";

	else if(f == r)
		cout<<cq[r]<<"\n";

	else if(r > f)
	{
		for(i = f; i <= r; i++)
		{
			cout<<cq[i]<<"\t";
		}
		cout<<"\n";
	}

	else
	{
		for(i = f; i <= size - 1; i++)
		{
			cout<<cq[i]<<"\t";
		}
		for(i = 0; i <= r; i++)
		{
			cout<<cq[i]<<"\t";
		}
		cout<<"\n";
	}

}


int main()
{
	int cq[size];
	int arr_even[size];
	int r = -1;
	int f = -1;

	insertion(cq, f, r);
	cout<<endl;
	print(cq, f, r);
	cout<<endl;
	con_Cq_arr(cq, arr_even, f, r);
	cout<<endl;
	print(cq, f, r);


    return 0;
}
```

Q3:) Having a circular Queue of size (10) , F=5 ,R=2 ,write program segment with draw to:
a) convert all values of circular queue to empty queue of size (7)?
b) Delete (3) elements from new queue?
c) Print the final state of Queue?

----

a/

```
#include <iostream>

using namespace std;

int const size1 = 10;
int const size2 = 7;

void con_C_Q(int cq1[size1], int &f1, int &r1, int cq2[size2], int &f2, int &r2)
{
      int i;
      int value;

      for(i = 0; i <= r1; i++)
      {
            ++r2;
            cq2[r2] = cq1[i];
      }

      if(f2 == -1)
            f2 = 0;

      for(i = f1; i <= size1 - 1; i++)
      {
            if(r2 == 6&& f2 == 0)
            {
                  value = cq2[f2];
                  ++f2;

                  r2 = 0;
                  cq2[r2] = cq1[i];

            }
            else
            {
                  ++r2;
                  cq2[r2] = cq1[i];
            }
      }
}

void printQ2(int cq2[size2], int f2, int r2)
{
      int i;

      for(i = f2; i <= size2 - 1; i++)
            cout<<cq2[i]<<"\t";

      for(i = 0; i == r2; i++)
            cout<<cq2[i]<<"\t";
      cout<<"\n";
}

int main()
{
      int cq1[size1] = {1, 2, 3, NULL, NULL, 6, 7, 8, 9, 10};
      int cq2[size2];

      int f1 = 5;
      int r1 = 2;
      int f2 = -1;
      int r2 = -1;

      cout<<"\n";
      con_C_Q(cq1, f1, r1, cq2, f2, r2);
      cout<<"\n";
      printQ2(cq2, f2, r2);
      cout<<"\n";

    return 0;
}
```

Q/Q1: Given a circular queue of size (8) contain (4) elements, Do the following (using program segment) :
A) add (5) elements? Then
B) Convert the negative values to empty stack of size (5)? Then
C) print the final state of the stack?

-

```
#include <iostream>

using namespace std;

int const size1 = 8;
int const size2 = 5;

void insertio(int cq[size1], int &f, int &r)
{
      int value;
      int i;

      for(i =0; i <= 4; i++)
      {
            if(r == 7 && f == 0)
            {
                  value =cq[f];
                  ++f;

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

                  r = 0;
                  cq[r] = value;
            }
            else
            {
                  cout<<"Enter any number : ";
                  cin>>value;

                  ++r;
                  cq[r] = value;
            }
      }
}

void co_s(int cq[size1], int stack[size2], int &top, int &f, int &r)
{
      int i;
      int value;

      for(i = f; i <= 7; i++)
      {
            if(cq[i] < 0)
            {
                  if(top == 4)
                  {
                        value = stack[top];
                        --top;
                        ++top;
                        stack[top] = cq[i];
                  }
                  else
                  {
                        ++top;
                        stack[top] = cq[i];
                  }
            }
      }
      for(i = 0; i == r; i++)
      {
            if(cq[i] < 0)
            {
                  if(top == 4)
                  {
                        value = stack[top];
                        --top;
                        ++top;
                        stack[top] = cq[i];
                  }
                  else
                  {
                        ++top;
                        stack[top] = cq[i];
                  }
            }
      }
}

void printQ(int cq[size1], int f, int r)
{
      int i;

      for(i = f; i <= 7; i++)
      {
            cout<<cq[i]<<"\t";
      }
      for(i = 0; i == r; i++)
      {
            cout<<cq[i]<<"\t";
      }
}

void prints(int stack[size2], int top)
{
      int i;

      for(i = 0; i <= top; i++)
      {
            cout<<stack[i]<<"\t";
      }
}


int main()
{
      int cq[size1] = {1, 2, -3, -4};
      int stack[size2];
      int f = 0;
      int r = 3;
      int top = -1;

      insertio(cq, f, r);
      cout<<"\n";

      co_s(cq, stack, top, f, r);
      cout<<"\n";

      printQ(cq, f, r);
      cout<<"\n";

      prints(stack, top);
      cout<<"\n";

    return 0;
}
```

Q1/ Given a circular queue of size (10) contain (10) elements, Convert the positive values to empty stack of size (10), and the negative values to empty Queue of size (10) , final print the stack and the queue ?

```
#include <iostream>

using namespace std;

int const size = 10;


void insertio(int cq[size], int &f1, int &r1)
{
      int value;
      int i;

      for(i =0; i <= 9; i++)
      {
            cout<<"Enter any number : ";
            cin>>value;

                  ++r1;
                  cq[r1] = value;
      }

      if(f1 == -1)
            f1 = 0;
}

void co_s(int cq[size], int stack[size], int &top, int &f1, int &r1, int
queue[size], int &f2, int &r2)
{
      int i;
      int value;

      for(i = f1; i <= r1; i++)
      {
            if(cq[i] >= 0)
            {
                  ++top;
                  stack[top] = cq[i];
            }

            else
            {
                  ++r2;
                  queue[r2] = cq[i];

                  if(f2 == -1)
                        f2 = 0;
            }
      }

}

void printcQ(int cq[size], int f1, int r1)
{
      int i;

      for(i = f1; i <= r1; i++)
      {
            cout<<cq[i]<<"\t";
      }
}

void prints(int stack[size], int top)
{
      int i;

      for(i = 0; i <= top; i++)
      {
            cout<<stack[i]<<"\t";
      }
}

void printq(int queue[size], int f2, int r2)
{
      int i;

      for(i = f2; i <= r2; i++)
      {
            cout<<queue[i]<<"\t";
      }
}


int main()
{
      int cq[size];
      int stack[size];
      int queue[size];

      int f1 = -1;
      int r1 = -1;
      int f2 = -1;
      int r2 = -1;
      int top = -1;

      insertio(cq, f1, r1);
      cout<<"\n";

      co_s(cq, stack, top, f1, r1, queue, f2, r2);
      cout<<"\nC Queue :\n";

      printcQ(cq, f1, r1);
      cout<<"\nStack :\n";

      prints(stack, top);
      cout<<"\nQueue :\n";

      printq(queue, f2, r2);
      cout<<"\n";

    return 0;
}
```

```
#include <iostream>

using namespace std;

int const size = 5;


void insertion(int cq[size], int &f, int &r)
{
      int value;
      int i;

      for(i = 0; i <= 4; i++)
      {
            if(r == size - 1 && f == 0)
                  cout<<"C Queue full !! insertion is not possible\n";

            else if(f - r == 1)
                  cout<<"C Queue full !! insertion is not possible\n";

            else if(r == size - 1 && f > 0)
            {
                  cout<<"Enter any value : ";
                  cin>>value;

                  r = 0;
                  cq[r] = value;
            }

            else
            {
                  cout<<"Enter any value : ";
                  cin>>value;

                  ++r;
                  cq[r] = value;
            }

            if(f == -1)
                  f = 0;
      }
}

void deletion(int cq[size], int &f, int &r)
{
      int value, i;

      for(i = 0; i <= 4; i++)
      {
            if(f == -1)
            cout<<"C Queue is empty !! is deletion is not possible \n";

            else if(f == r)
            {
                  value = cq[f];
                  f = -1;
                  r = -1;
            }
            else if(f == size - 1 && f > r)
            {
                  value = cq[f];
                  f = 0;
            }

            else
            {
                  value = cq[f];
                  ++f;
            }
            }

}

void print(int cq[size], int &f, int &r)
{
      int i;

      if(f == -1)
            cout<<"C Queue is empty\n";

      else if(f == r)
            cout<<cq[r]<<"\n";

      else if(r > f)
      {
            for(i = f; i <= r; i++)
            {
                  cout<<cq[i]<<"\t";
            }
            cout<<"\n";
      }

      else
      {
            for(i = f; i <= size - 1; i++)
            {
                  cout<<cq[i]<<"\t";
            }
            for(i = 0; i <= r; i++)
            {
                  cout<<cq[i]<<"\t";
            }
            cout<<"\n";
      }

}


int main()
{
      int cq[size];
      int r = -1;
      int f = -1;

      insertion(cq, f, r);
      cout<<endl;
      print(cq, f, r);
```

```
        cout<<endl;
        deletion(cq, f, r);
        print(cq, f, r);


    return 0;
}
```

```
#include <iostream>

using namespace std;

int const size = 6;

void insertion(int cq[size], int &f, int &r)
{
      int value;

      if(r == size - 1 && f == 0)
            cout<<"C Queue full !! insertion is not possible\n";
      else if(f - r == 1)
            cout<<"C Queue full !! insertion is not possible\n";
      else if(r == size - 1 && f > 0)
      {
            cout<<"Enter any value : ";
            cin>>value;

            r = 0;
            cq[r] = value;
      }
      else
      {
            cout<<"Enter any value : ";
            cin>>value;

            ++r;
            cq[r] = value;
      }

      if(f == -1)
            f = 0;
}

void deletion(int cq[size], int &f, int &r)
{
      int value;

      if(f == -1)
            cout<<"C Queue is empty !! is deletion is not possible \n";

      else if(f == r)
      {
            value = cq[f];
            f = -1;
            r = -1;
      }

      else if(f == size - 1 && f > r)
      {
            value = cq[f];
            f = 0;
      }

      else
      {
            value = cq[f];
            ++f;
      }

}

void print(int cq[size], int &f, int &r)
{
      int i;

      if(f == -1)
            cout<<"C Queue is empty\n";

      else if(f == r)
            cout<<cq[r]<<"\n";

      else if(r > f)
      {
            for(i = f; i <= r; i++)
            {
                  cout<<cq[i]<<"\t";
            }
            cout<<"\n";
      }

      else
      {
            for(i = f; i <= size - 1; i++)
            {
                  cout<<cq[i]<<"\t";
            }
            for(i = 0; i <= r; i++)
            {
                  cout<<cq[i]<<"\t";
            }
            cout<<"\n";
      }

}


int main()
{
      int cq[size];
      int r = -1;
      int f = -1;
      int x;

      for( ; x != 4; )
      {
            cout<<"1- insert \n";
            cout<<"2- delete \n";
            cout<<"3- print \n";
            cout<<"4- exit \n";

            cout<<"Enter your choose :";
            cin>>x;

            switch(x)
            {
                  case 1: insertion(cq, f, r); break;
```

```
                case 2: deletion(cq, f, r); break;
                case 3: print(cq, f, r); break;
                default: cout<<"Error\n";
            }
        }

    return 0;
}
```

```
#include<iostream>
using namespace std;

struct node
{
      int info;
      struct node* next;
}; typedef struct node* nodeptr;

void insertbegin(nodeptr &plist)
{
      nodeptr p;
      p = new node;

      cout<<"Enter any number : ";
      cin>>p->info;

      p->next = plist;
      plist = p;
}

void insertend(nodeptr &plist)
{
      nodeptr q, p;
      q = plist;

      for( ; q->next != NULL; )
            q = q->next;

      p = new node;

      cout<<"Enter any number : ";
      cin>>p->info;

      p->next = NULL;
      q->next = p;
}

void ins_betwen(nodeptr &plist)
{
    nodeptr p, a, b;
    int i, L;

    cout << "Enter the number of location : ";
    cin >> L;

    p = new node;

    cout << "Enter any number : ";
    cin >> p->info;

    a = plist;

    for (i = 2; i < L; i++)
        a = a->next;

    b = a->next;
    a->next = p;
    p->next = b;
```

```
}

void deletebeg(nodeptr &plist)
{
      nodeptr p;

      p = plist;
      plist = plist->next;
      free(p);
}

void deleteend(nodeptr &plist)
{
      nodeptr q, p;
      q = plist;
      p = plist;

      for( ; q->next != NULL; )
      {
            p = q;
            q = q->next;
      }
      p->next = NULL;
      free(q);
}

void deletebet(nodeptr &plist)
{
      nodeptr p, q, a;
      int i, l;
      p = plist;
      q = plist;
      a = plist;

      cout << "Enter the number of location : ";
    cin >> l;

      for(i = 2; i < l; i++)
      {
            p = q;
            q = q->next;
            a = q->next;
      }

      q->next = a->next;

      free(a);
}

void displaying(nodeptr &plist)
{
      nodeptr q;

      if(plist == NULL)
            cout<<"\nList is Empty\n";
      else
      {
            q = plist;
            for( ; q != NULL; )
            {
                  cout<<q->info<<"\t";
                  q = q->next;
            }
            cout<<"\n";
      }
}

int main()
{
      nodeptr p, q, plist;
      int x;
      p = new node;

      cout<<"Enter any number : ";
      cin>>p->info;
      p->next = NULL;
      plist = p;

      cout<<endl;
      for( ; x != 8; )
      {
            cout<<"1- add from begin\n";
            cout<<"2- add from end\n";
            cout<<"3- add from between\n";
            cout<<"4- delete from begin\n";
            cout<<"5- delete from end\n";
            cout<<"6- delete from between\n";
            cout<<"7- print list\n";
            cout<<"8- exit\n";

            cout<<"Enter your choise : ";
            cin>>x;

            cout<<endl;
            switch(x)
            {
                  case 1:insertbegin(plist); break;
                  case 2:insertend(plist); break;
                  case 3:ins_betwen(plist); break;
                  case 4:deletebeg(plist); break;
                  case 5:deleteend(plist); break;
                  case 6:deletebet(plist); break;
                  case 7:displaying(plist); break;
                  default:cout<<"Error\n";
            }
      }

      return 0;
}
```

```
#include<iostream>
using namespace std;

int const size = 1;

struct node
{
      int info;
      struct node* next;
}; typedef struct node* nodeptr;

void insertbegin(nodeptr &plist)
{
      nodeptr p;
      p = new node;

      cout<<"Enter any number : ";
      cin>>p->info;

      p->next = plist;
      plist = p;
}

void insertend(nodeptr &plist)
{
      nodeptr q, p;
      q = plist;

      for( ; q->next != NULL; )
            q = q->next;

      p = new node;

      cout<<"Enter any number : ";
      cin>>p->info;

      p->next = NULL;
      q->next = p;
}

void ins_betwen(nodeptr &plist)
{
    nodeptr p, a, b;
    int i, L;

    cout << "Enter the number of location : ";
    cin >> L;

    p = new node;

    cout << "Enter any number : ";
    cin >> p->info;

    a = plist;

    for (i = 2; i < L; i++)
        a = a->next;

    b = a->next;
    a->next = p;
    p->next = b;
}

void deletebeg(nodeptr &plist)
{
      nodeptr p;

      p = plist;
      plist = plist->next;
      free(p);
}

void deleteend(nodeptr &plist)
{
      nodeptr q, p;
      q = plist;
      p = plist;

      for( ; q->next != NULL; )
      {
            p = q;
            q = q->next;
      }
      p->next = NULL;
      free(q);
}

void deletebet(nodeptr &plist)
{
      nodeptr p, q, a;
      int i, l;
      p = plist;
      q = plist;
      a = plist;

      cout << "Enter the number of location : ";
    cin >> l;

      for(i = 2; i < l; i++)
      {
            p = q;
            q = q->next;
            a = q->next;
      }

      q->next = a->next;

      free(a);
}

void prime(nodeptr &plist)
{
      nodeptr q;
      int x, i, j=1, l = 0;

      q = plist;

      for( ; q != NULL; )
```

```
	{
		if(q->info != 1)
		{
			x = 0;
			for(i = 2; i < q->info; i++)
			{
				if(q->info % 2 == 0)
					x = 1;
			}

			if(x == 0)
			{
				cout<<q->info<<"\t";
				++l;
			}
		}

		else
		{
			cout<<q->info<<"\t";
			++l;
		}

		q = q->next;
	}

	cout<<endl<<"Number numbers prime :"<<l<<endl<<endl;
}

void sum_num(nodeptr &plist)
{
	nodeptr q;
	int sum = 0, i = 0;

	q = plist;

	for( ; q != NULL; )
	{
		if(q->info % 5 == 0)
		{
			sum += q->info;
			++i;
		}
		q = q->next;
	}
	cout<<"Sum number %5 == 0 : "<<sum<<"\n";
	cout<<"Number numbers %5 == 0 : "<<i<<"\n";
}

void con_arr(nodeptr &plist, int array[size], int &y)
{
	nodeptr q;
	int i = 0, z = 0;
	q = plist;


	for( ; q != NULL; )
	{
		if(q->info % 2 == 0)
```

```
            {
                  ++z;
                  if(z > 1)
                        ++y;

                  array[i] = q->info;
                  ++i;
            }
            q = q->next;
      }

      for(i = 0; i <= y; i++)
            cout<<array[i]<<"\t";
      cout<<endl;
}

void fact(nodeptr &plist)
{
      nodeptr q;
      int i, fact = 1, maxamimm;

      q = plist;

      maxamimm = q->info;

      for( ; q != NULL; )
      {
            if(maxamimm < q->info)
                  maxamimm = q->info;

            q = q->next;
      }

      for(i = 1; i <= maxamimm; i++)
      {
            fact = fact * i;
      }

      cout<<"Fact ( "<<maxamimm<<" ) : "<<fact<<"\n";
}

void number(nodeptr &plist)
{
      nodeptr q;
      int i = 0;
      q = plist;

      for( ; q != NULL; )
      {
            ++i;
            q = q->next;
      }

      cout<<"number nods : "<<i<<"\n\n";
}

void displaying(nodeptr &plist)
{
      nodeptr q;
```

```
	if(plist == NULL)
		cout<<"\nList is Empty\n";
	else
	{
		q = plist;

		for( ; q != NULL; )
		{
			cout<<q->info<<"\t";
			q = q->next;
		}
		cout<<"\n";
	}
}

int main()
{
	nodeptr p, q, plist;
	int x, y = 0;
	int array[size + y];
	p = new node;

	cout<<"Enter any number : ";
	cin>>p->info;
	p->next = NULL;
	plist = p;

	cout<<endl;
	for( ; x != 13; )
	{
		cout<<"1- add from begin\n";
		cout<<"2- add from end\n";
		cout<<"3- add from between\n";
		cout<<"4- delete from begin\n";
		cout<<"5- delete from end\n";
		cout<<"6- delete from between\n";
		cout<<"7- prime list elem\n";
		cout<<"8- Sum number %5 == 0 and number him list elem\n";
		cout<<"9- convert even to array\n";
		cout<<"10- fact maximamm list\n";
		cout<<"11- numbers node in list\n";
		cout<<"12- print list\n";
		cout<<"13- exit\n";

		cout<<"Enter your choise : ";
		cin>>x;

		cout<<endl;
		switch(x)
		{
			case 1:insertbegin(plist); break;
			case 2:insertend(plist); break;
			case 3:ins_betwen(plist); break;
			case 4:deletebeg(plist); break;
			case 5:deleteend(plist); break;
			case 6:deletebet(plist); break;
			case 7:prime(plist); break;
			case 8:sum_num(plist); break;
```

```
                  case 9:con_arr(plist, array, y); break;
                  case 10:fact(plist); break;
                  case 11:number(plist); break;
                  case 12:displaying(plist); break;
                  default:cout<<"Error\n";
            }
      }

      return 0;
}
```

```
#include<iostream>
using namespace std;

int const size = 1;

struct node
{
      int info;
      struct node* next;
}; typedef struct node* nodeptr;


void insertend(nodeptr &plist)
{
      nodeptr q, p;
      int i;
      q = plist;

      for(i = 0; i <= 12; i++)
      {
            for( ; q->next != NULL; )
            q = q->next;

            p = new node;

            cout<<"Enter any number : ";
            cin>>p->info;

            p->next = NULL;
            q->next = p;
      }

}

void spilt(nodeptr &plist)
{
      nodeptr q, p, head1, head2, a, b, c;
      int i;
      a = plist;

      cout<<"List 1 : \n";
      p = new node;
      p->info = a->info;
      p->next = NULL;
      head1 = p;
      b = head1;
      cout<<p->info<<"\t";

      a = a->next;
      for(i = 0; i <= 5; i++)
      {
            for( ; b->next != NULL; )
                        b = b->next;

            c = new node;
            c->info = a->info;
            cout<<c->info<<"\t";
            c->next = NULL;
            b->next = c;
            a = a->next;
      }

      cout<<"\nList 2 : \n";
      nodeptr b1, c1;
      b1 = head2;

      q = new node;
      q->info = a->info;
      q->next = NULL;
      head2 = q;
      cout<<q->info<<"\t";

      a = a->next;

      for(i = 7; i <= 12; i++)
      {
            for( ; b1->next != NULL; )
                        b1 = b1->next;

            c1 = new node;
            c1->info = a->info;
            cout<<c1->info<<"\t";
            c1->next = NULL;
            b1->next = c1;

            a = a->next;
      }


}

void displaying(nodeptr &plist)
{
      nodeptr q;

      if(plist == NULL)
            cout<<"\nList is Empty\n";
      else
      {
            q = plist;

            for( ; q != NULL; )
            {
                  cout<<q->info<<"\t";
                  q = q->next;
            }
            cout<<"\n";
      }
}

int main()
{
      nodeptr p, q, plist;
      int x, y = 0;
      int array[size + y];
      p = new node;
```

```
	cout<<"Enter any number : ";
	cin>>p->info;
	p->next = NULL;
	plist = p;

	cout<<endl;
	insertend(plist);

	cout<<endl;
	displaying(plist);

	cout<<endl;
	spilt(plist);


	return 0;
}
```