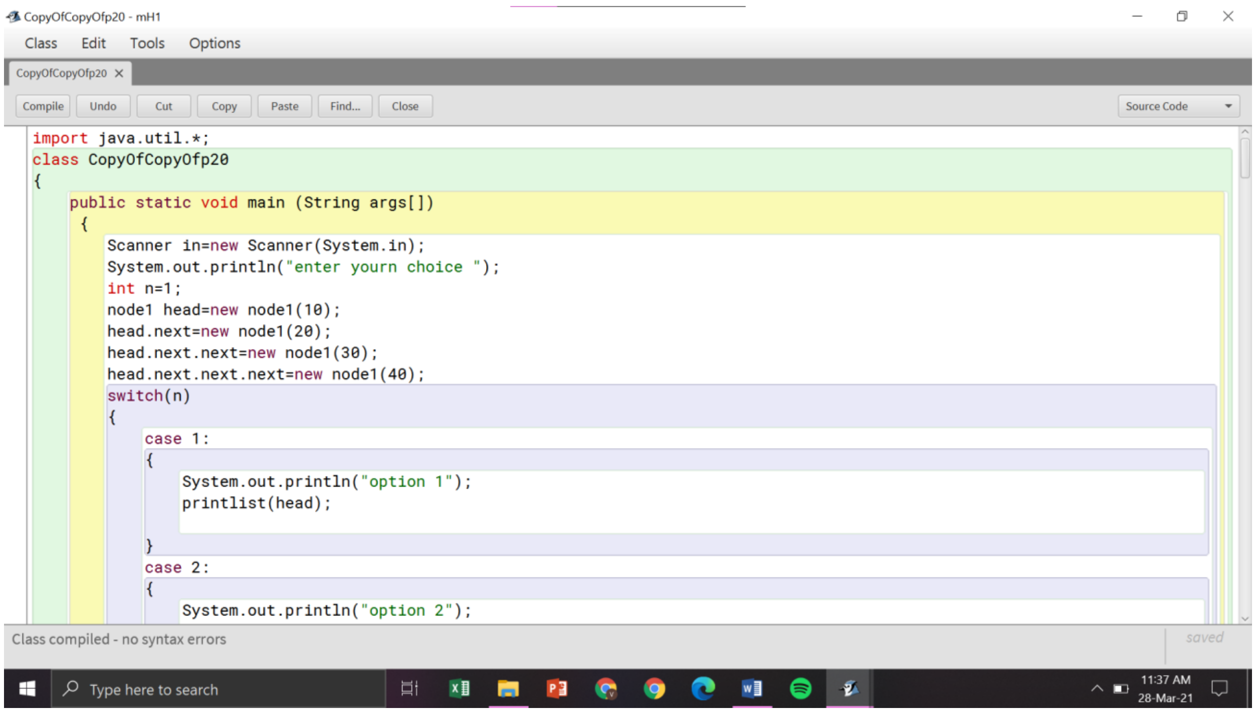
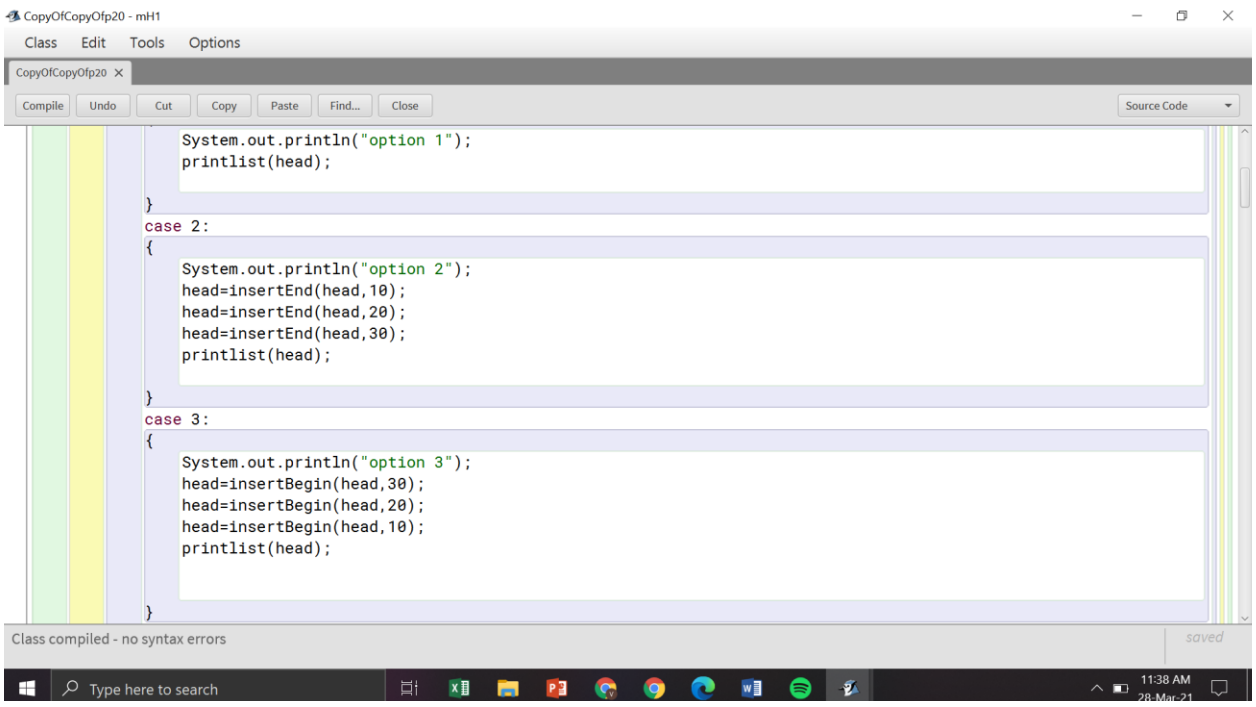

CopyOfCopyOfp20 - mH1

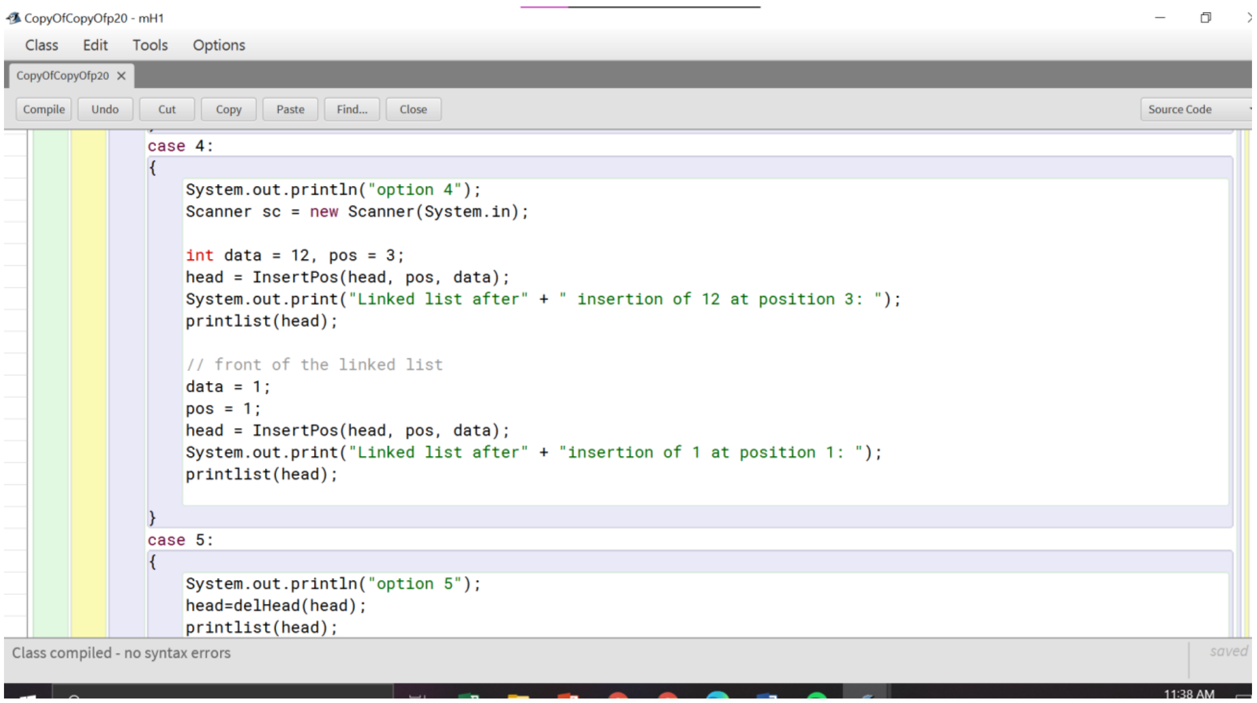
```
import java.util.*;
class CopyOfCopyOfp20
{
    public static void main (String args[])
     {
        Scanner in=new Scanner(System.in);
        System.out.println("enter yourn choice ");
        int n=1;
        node1 head=new node1(10);
        head.next=new node1(20);
        head.next.next=new node1(30);
        head.next.next.next=new node1(40);
        switch(n)
        {
            case 1:
            {
                System.out.println("option 1");
                printlist(head);

            }
            case 2:
            {
                System.out.println("option 2");
```

Class compiled - no syntax errors


CopyOfCopyOfp20 - mH1

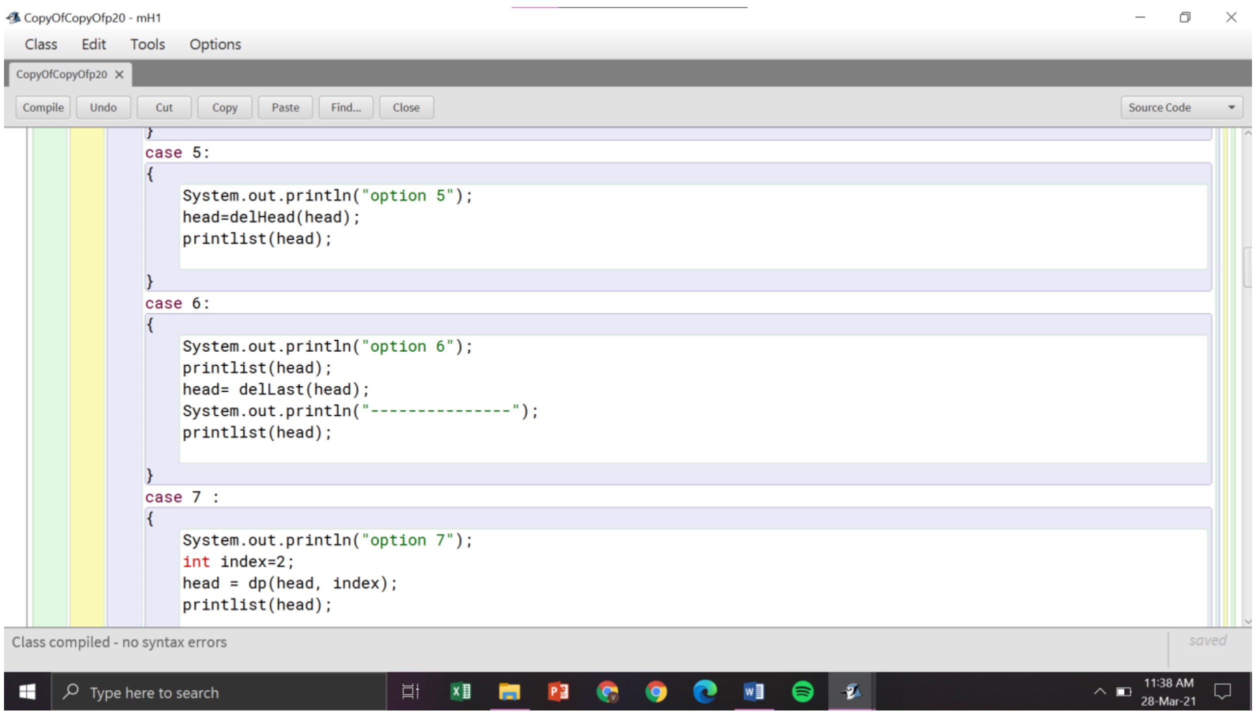
```
                System.out.println("option 1");
                printlist(head);

            }
            case 2:
            {
                System.out.println("option 2");
                head=insertEnd(head,10);
                head=insertEnd(head,20);
                head=insertEnd(head,30);
                printlist(head);

            }
            case 3:
            {
                System.out.println("option 3");
                head=insertBegin(head,30);
                head=insertBegin(head,20);
                head=insertBegin(head,10);
                printlist(head);


            }
```

Class compiled - no syntax errors



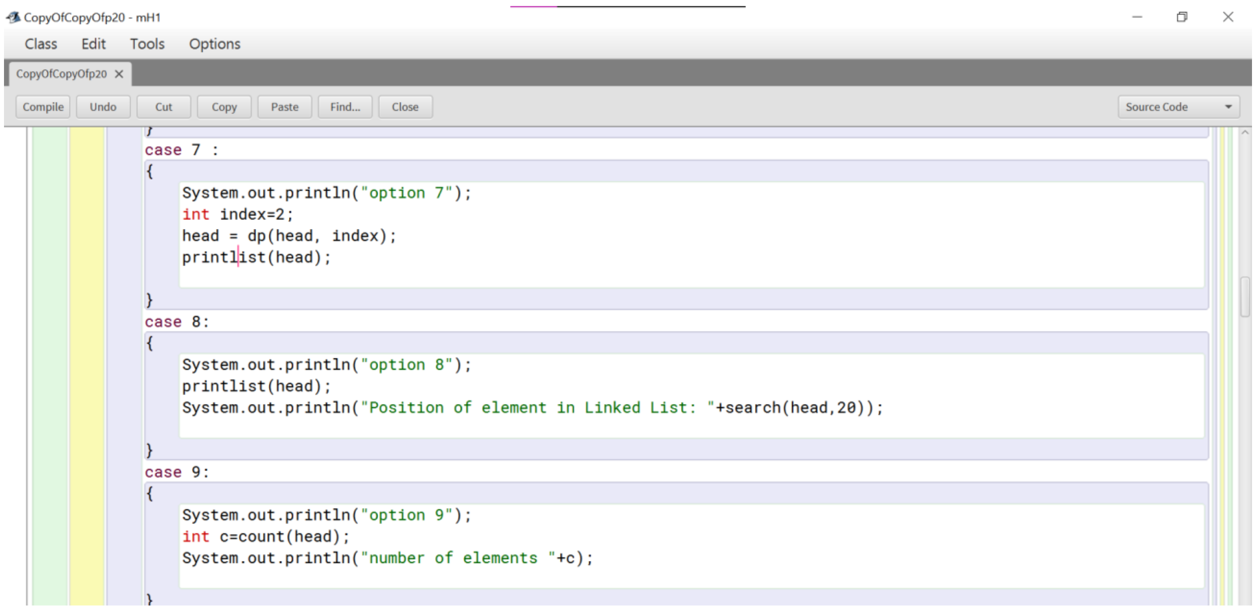
```
case 4:
{
    System.out.println("option 4");
    Scanner sc = new Scanner(System.in);

    int data = 12, pos = 3;
    head = InsertPos(head, pos, data);
    System.out.print("Linked list after" + " insertion of 12 at position 3: ");
    printlist(head);

    // front of the linked list
    data = 1;
    pos = 1;
    head = InsertPos(head, pos, data);
    System.out.print("Linked list after" + "insertion of 1 at position 1: ");
    printlist(head);

}
case 5:
{
    System.out.println("option 5");
    head=delHead(head);
    printlist(head);
```

Class compiled - no syntax errors

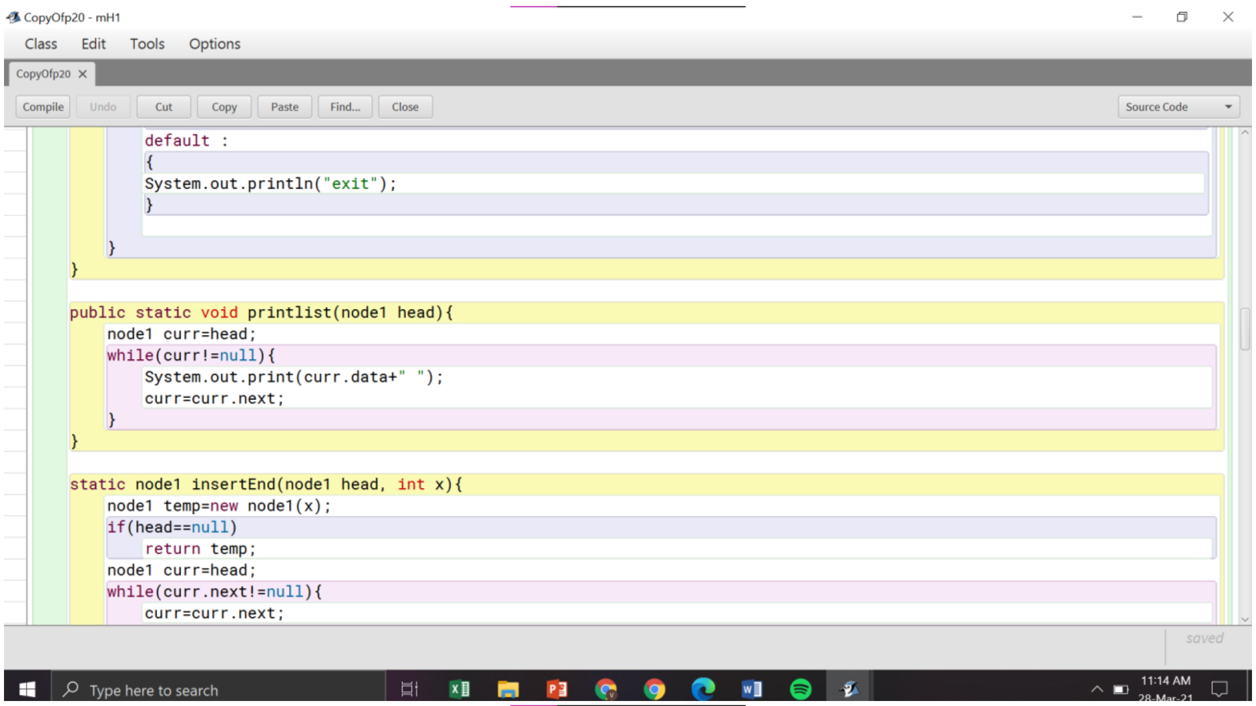


```
}
case 5:
{
    System.out.println("option 5");
    head=delHead(head);
    printlist(head);

}
case 6:
{
    System.out.println("option 6");
    printlist(head);
    head= delLast(head);
    System.out.println("---------------");
    printlist(head);

}
case 7 :
{
    System.out.println("option 7");
    int index=2;
    head = dp(head, index);
    printlist(head);
```

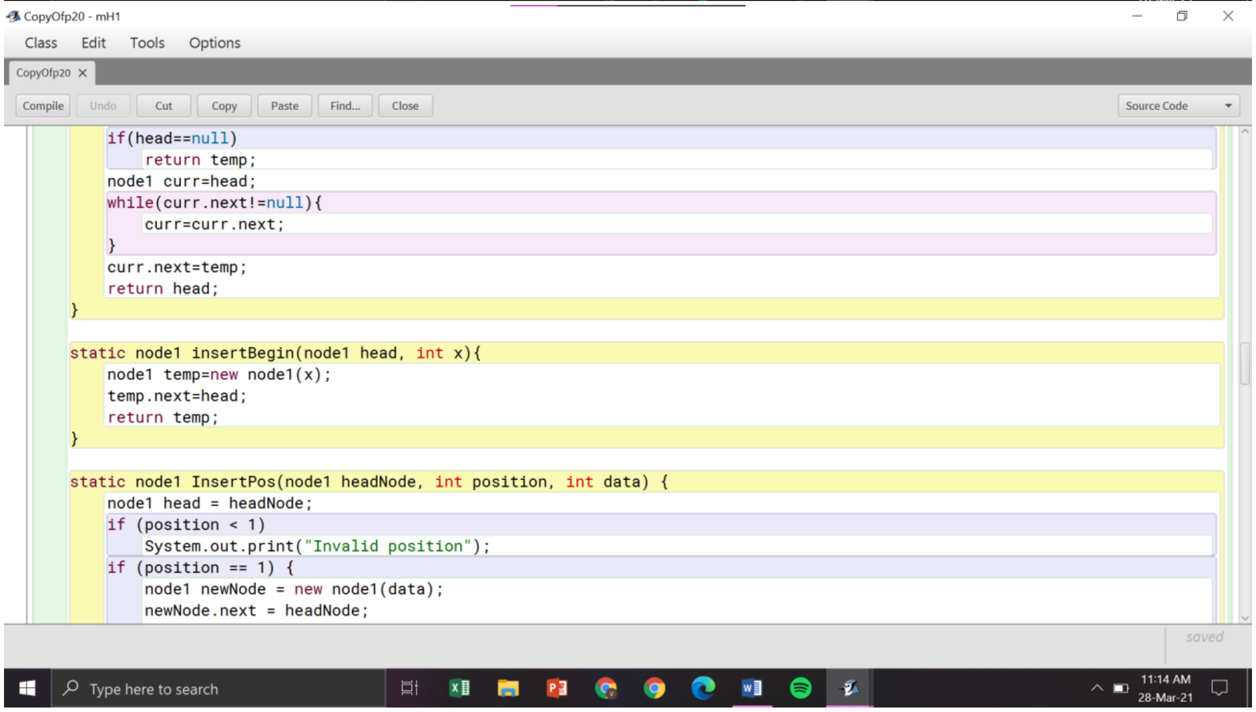
Class compiled - no syntax errors



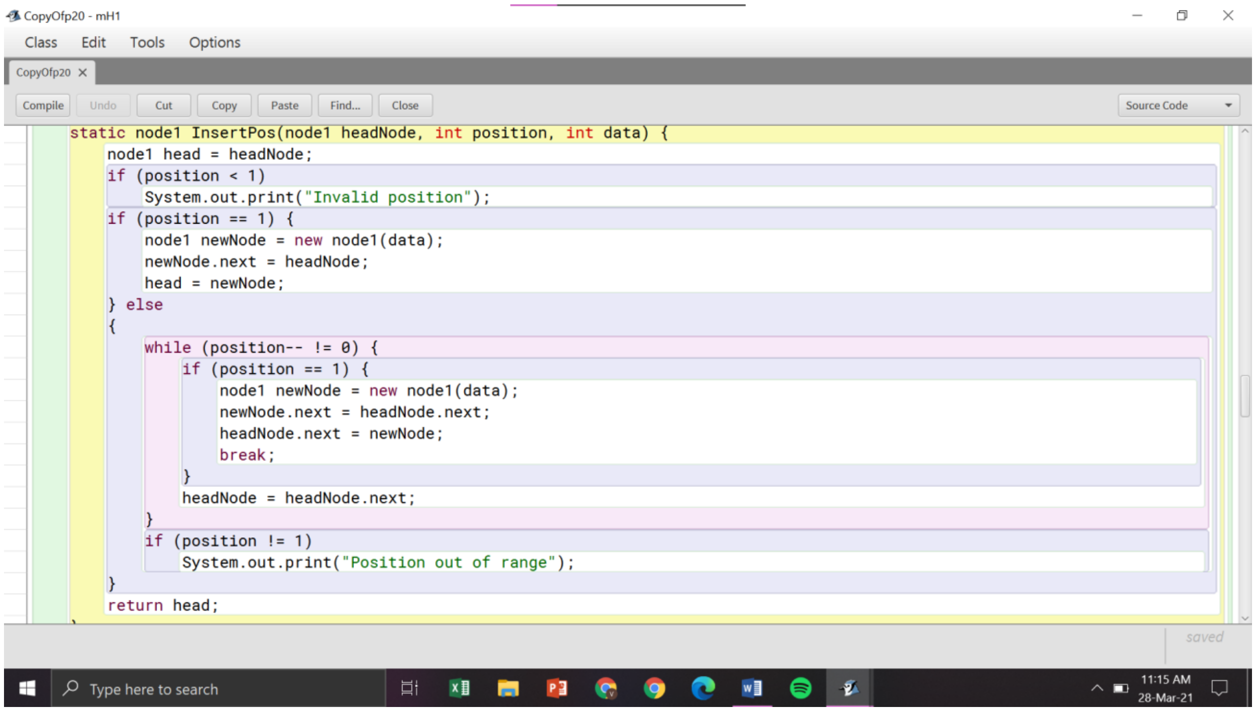
```
case 7 :
{
    System.out.println("option 7");
    int index=2;
    head = dp(head, index);
    printlist(head);

}
case 8:
{
    System.out.println("option 8");
    printlist(head);
    System.out.println("Position of element in Linked List: "+search(head,20));

}
case 9:
{
    System.out.println("option 9");
    int c=count(head);
    System.out.println("number of elements "+c);

}
```



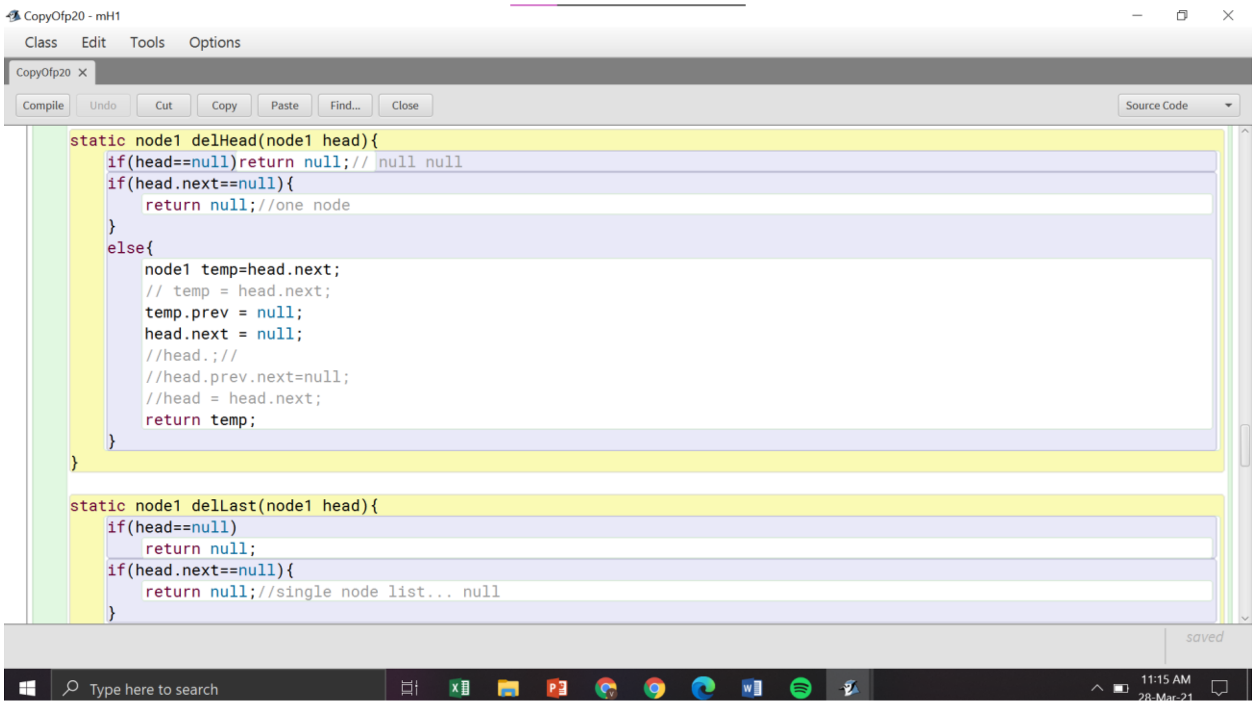
```
            default :
            {
            System.out.println("exit");
            }

        }
}

public static void printlist(node1 head){
    node1 curr=head;
    while(curr!=null){
        System.out.print(curr.data+" ");
        curr=curr.next;
    }
}

static node1 insertEnd(node1 head, int x){
    node1 temp=new node1(x);
    if(head==null)
        return temp;
    node1 curr=head;
    while(curr.next!=null){
        curr=curr.next;
```



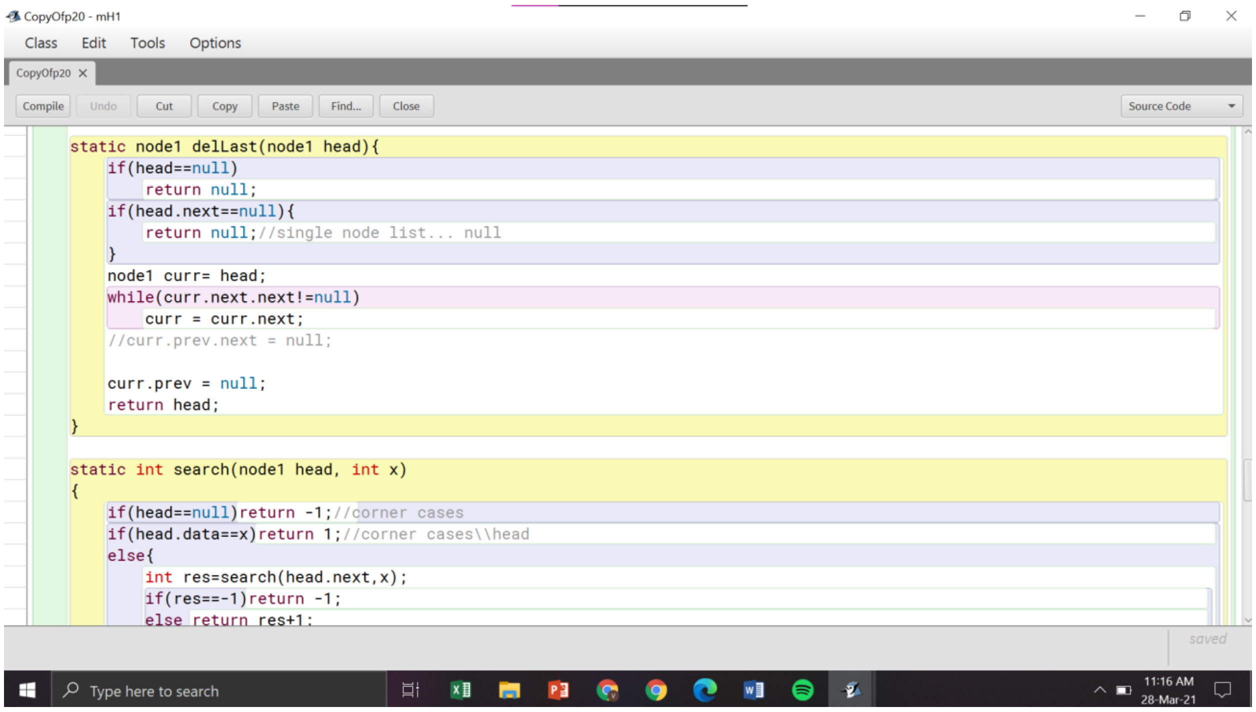
```
    if(head==null)
        return temp;
    node1 curr=head;
    while(curr.next!=null){
        curr=curr.next;
    }
    curr.next=temp;
    return head;
}

static node1 insertBegin(node1 head, int x){
    node1 temp=new node1(x);
    temp.next=head;
    return temp;
}

static node1 InsertPos(node1 headNode, int position, int data) {
    node1 head = headNode;
    if (position < 1)
        System.out.print("Invalid position");
    if (position == 1) {
        node1 newNode = new node1(data);
        newNode.next = headNode;
```



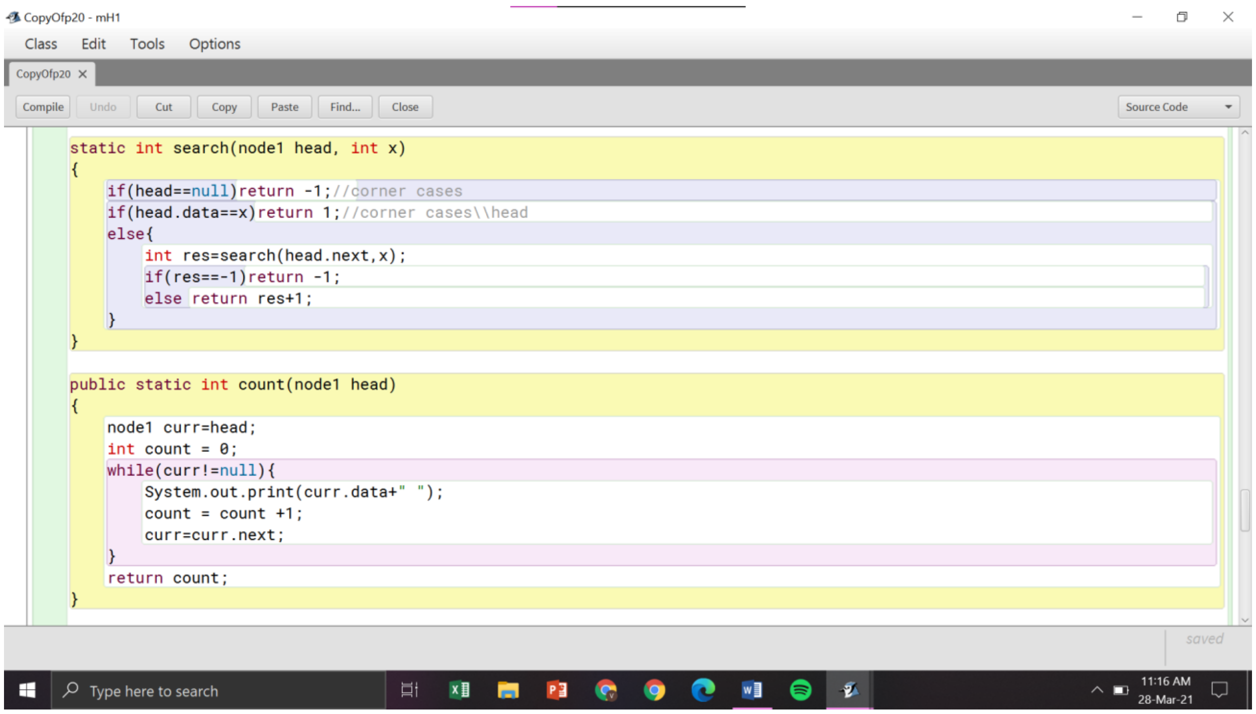
```java
static node1 InsertPos(node1 headNode, int position, int data) {
    node1 head = headNode;
    if (position < 1)
        System.out.print("Invalid position");
    if (position == 1) {
        node1 newNode = new node1(data);
        newNode.next = headNode;
        head = newNode;
    } else
    {
        while (position-- != 0) {
            if (position == 1) {
                node1 newNode = new node1(data);
                newNode.next = headNode.next;
                headNode.next = newNode;
                break;
            }
            headNode = headNode.next;
        }
        if (position != 1)
            System.out.print("Position out of range");
    }
    return head;
```



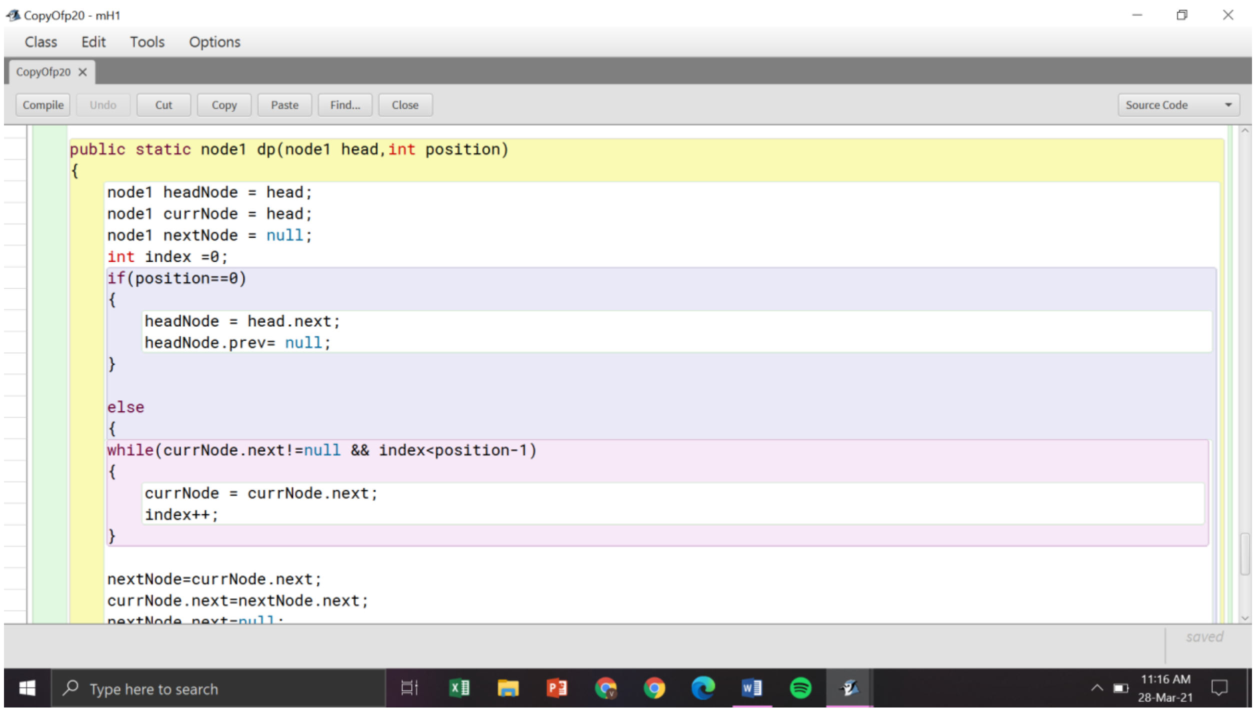
```java
static node1 delHead(node1 head){
    if(head==null)return null;// null null
    if(head.next==null){
        return null;//one node
    }
    else{
        node1 temp=head.next;
        // temp = head.next;
        temp.prev = null;
        head.next = null;
        //head.;//
        //head.prev.next=null;
        //head = head.next;
        return temp;
    }
}

static node1 delLast(node1 head){
    if(head==null)
        return null;
    if(head.next==null){
        return null;//single node list... null
    }
```



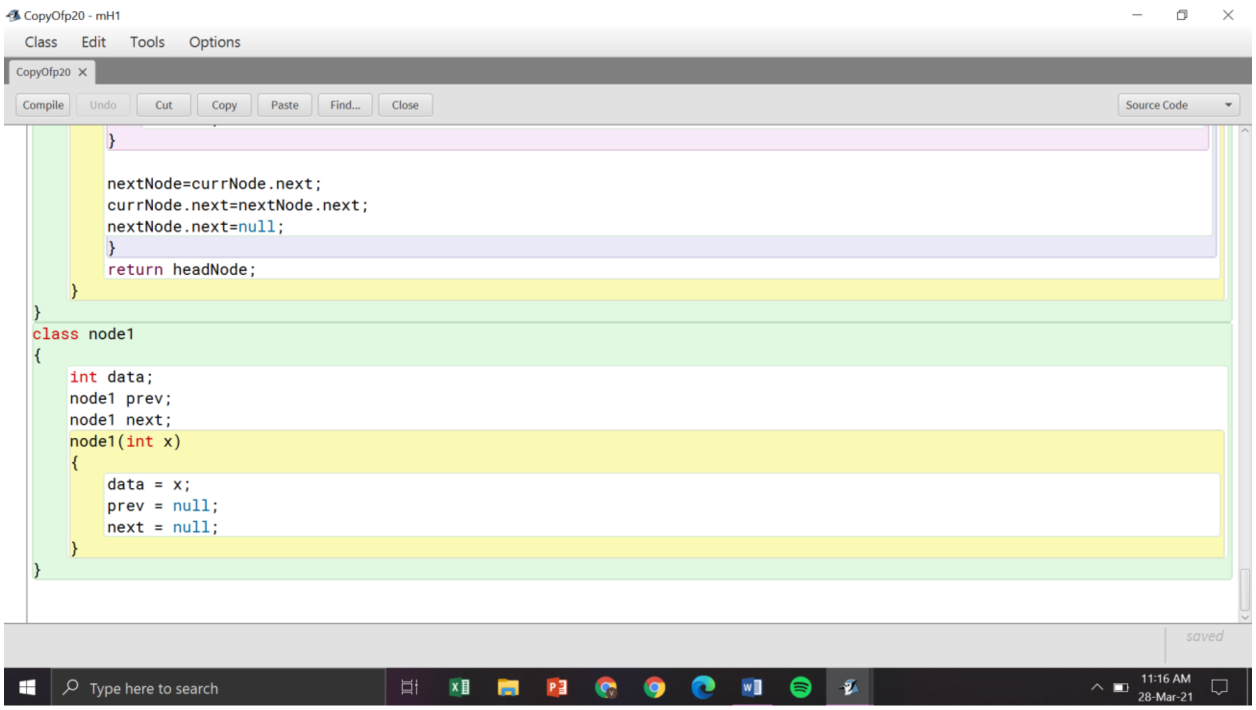
```
static node1 delLast(node1 head){
    if(head==null)
        return null;
    if(head.next==null){
        return null;//single node list... null
    }
    node1 curr= head;
    while(curr.next.next!=null)
        curr = curr.next;
    //curr.prev.next = null;

    curr.prev = null;
    return head;
}

static int search(node1 head, int x)
{
    if(head==null)return -1;//corner cases
    if(head.data==x)return 1;//corner cases\\head
    else{
        int res=search(head.next,x);
        if(res==-1)return -1;
        else return res+1;
```



```
static int search(node1 head, int x)
{
    if(head==null)return -1;//corner cases
    if(head.data==x)return 1;//corner cases\\head
    else{
        int res=search(head.next,x);
        if(res==-1)return -1;
        else return res+1;
    }
}

public static int count(node1 head)
{
    node1 curr=head;
    int count = 0;
    while(curr!=null){
        System.out.print(curr.data+" ");
        count = count +1;
        curr=curr.next;
    }
    return count;
}
```

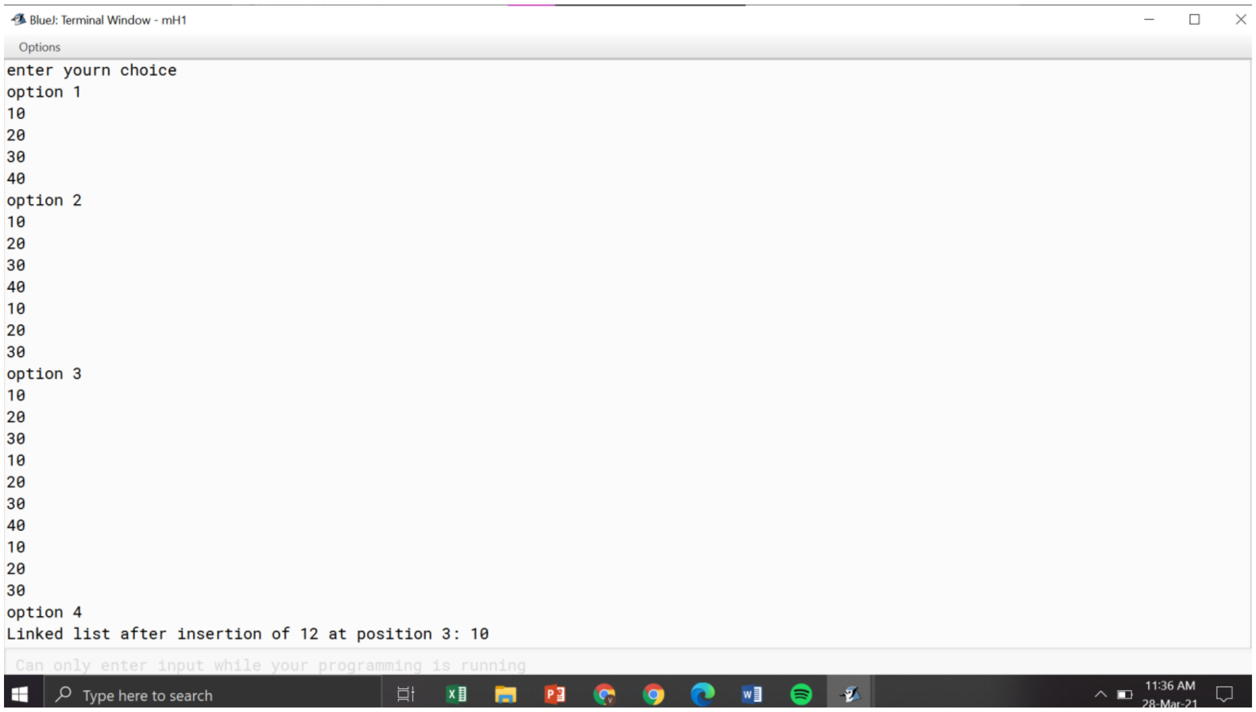


```
public static node1 dp(node1 head,int position)
{
    node1 headNode = head;
    node1 currNode = head;
    node1 nextNode = null;
    int index =0;
    if(position==0)
    {
        headNode = head.next;
        headNode.prev= null;
    }

    else
    {
    while(currNode.next!=null && index<position-1)
    {
        currNode = currNode.next;
        index++;
    }

    nextNode=currNode.next;
    currNode.next=nextNode.next;
```



```
    }

    nextNode=currNode.next;
    currNode.next=nextNode.next;
    nextNode.next=null;
    }
    return headNode;
}
}
class node1
{
    int data;
    node1 prev;
    node1 next;
    node1(int x)
    {
        data = x;
        prev = null;
        next = null;
    }
}
```

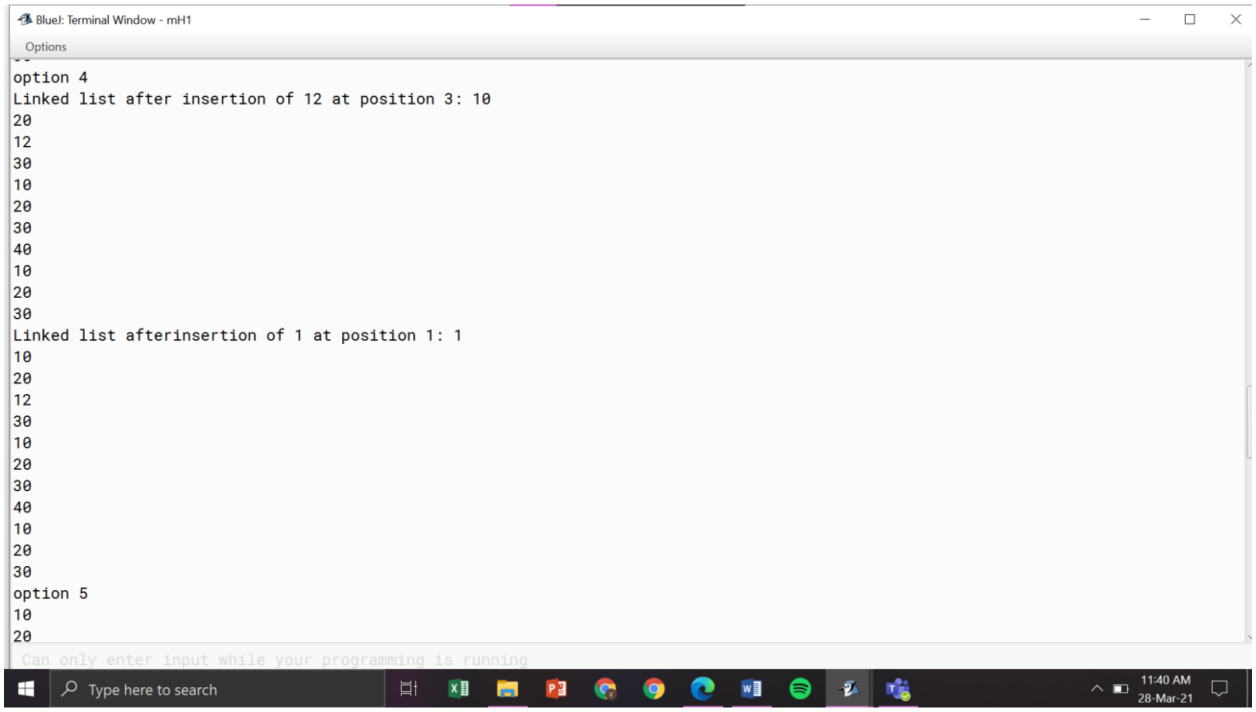
Output :


BlueJ: Terminal Window - mH1

Options

```
enter yourn choice
option 1
10
20
30
40
option 2
10
20
30
40
10
20
30
option 3
10
20
30
10
20
30
40
10
20
30
option 4
Linked list after insertion of 12 at position 3: 10
```

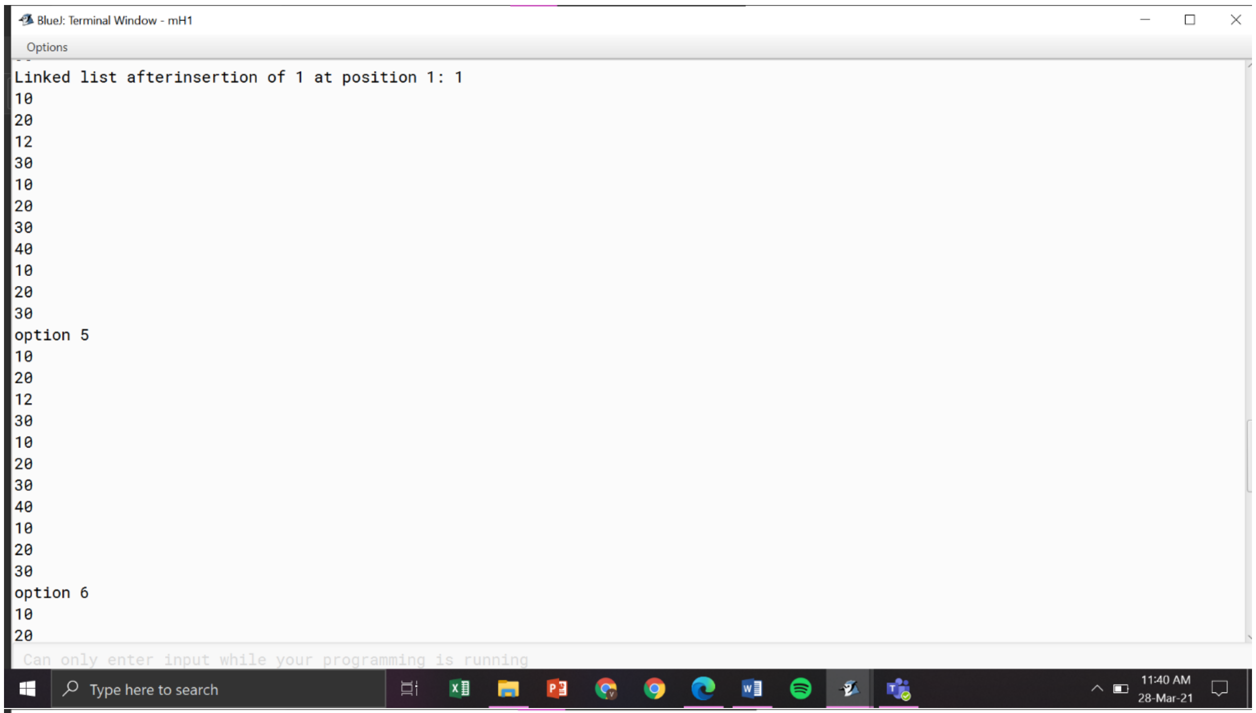
Can only enter input while your programming is running


BlueJ: Terminal Window - mH1

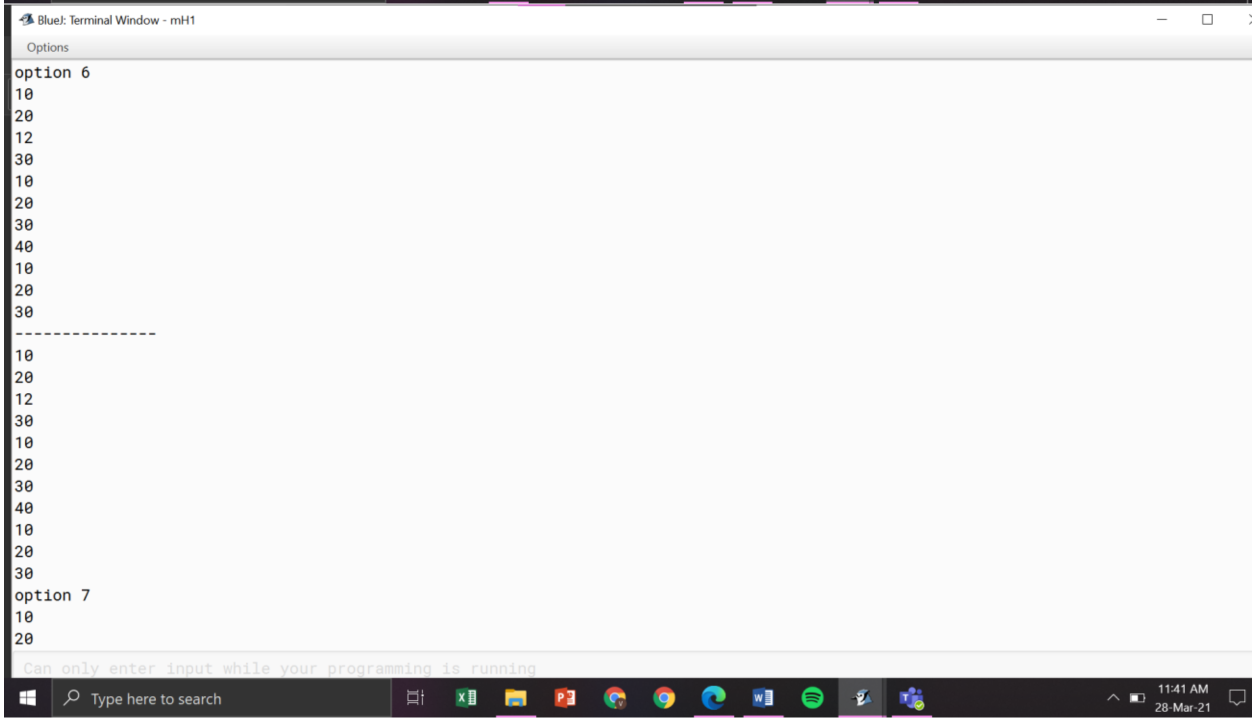
Options

```
option 4
Linked list after insertion of 12 at position 3: 10
20
12
30
10
20
30
40
10
20
30
Linked list afterinsertion of 1 at position 1: 1
10
20
12
30
10
20
30
40
10
20
30
option 5
10
20
```

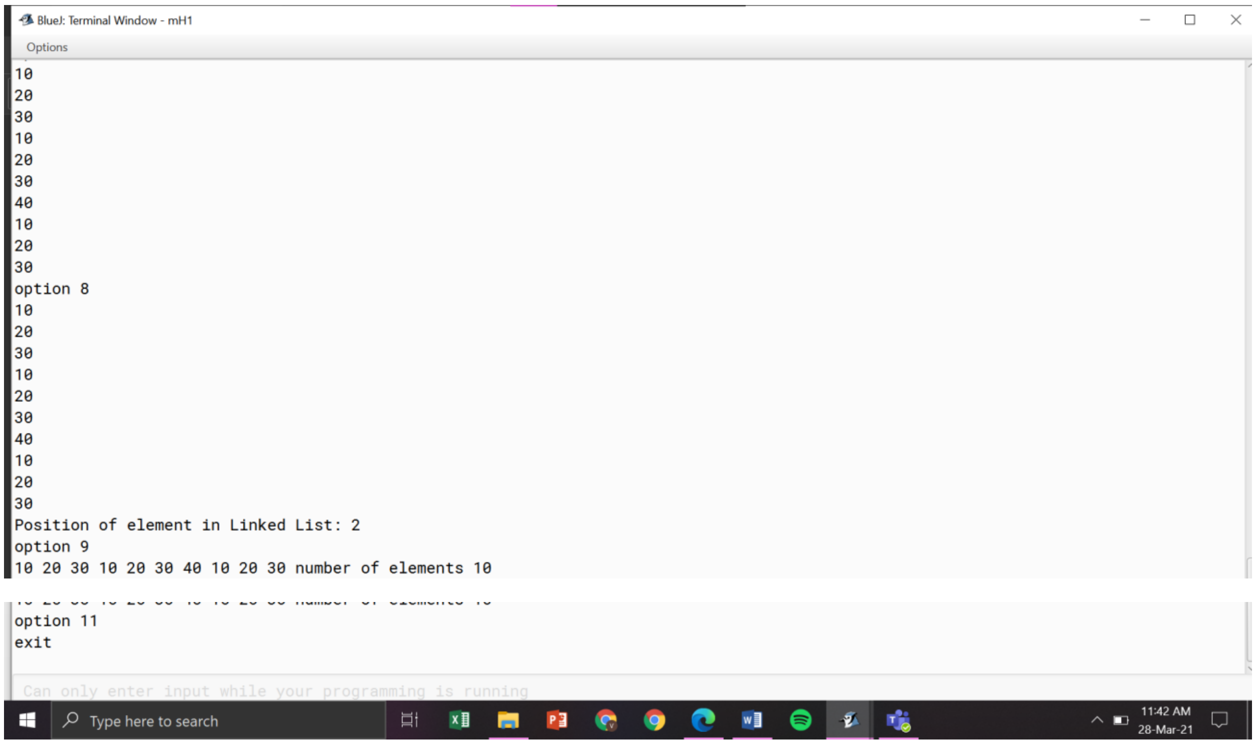
Can only enter input while your programming is running


BlueJ: Terminal Window - mH1

```
Linked list afterinsertion of 1 at position 1: 1
10
20
12
30
10
20
30
40
10
20
30
option 5
10
20
12
30
10
20
30
40
10
20
30
option 6
10
20
```


BlueJ: Terminal Window - mH1

```
option 6
10
20
12
30
10
20
30
40
10
20
30
---------------
10
20
12
30
10
20
30
40
10
20
30
option 7
10
20
```


BlueJ: Terminal Window - mH1
Options

```
10
20
30
10
20
30
40
10
20
30
option 8
10
20
30
10
20
30
40
10
20
30
Position of element in Linked List: 2
option 9
10 20 30 10 20 30 40 10 20 30 number of elements 10
option 11
exit
```

Can only enter input while your programming is running
Type here to search
11:42 AM
28-Mar-21

Code :

```
import java.util.*;

class CopyOfp20

{

  public static void main (String args[])

   {

     Scanner in=new Scanner(System.in);

     System.out.println("enter yourn choice ");

     int n=in.nextInt();

     node1 head=new node1(10);

     head.next=new node1(20);

     head.next.next=new node1(30);

     head.next.next.next=new node1(40);

     switch(n)

     {
```

```
case 1:
{
    printlist(head);
    break;
}
case 2:
{
    head=insertEnd(head,10);
    head=insertEnd(head,20);
    head=insertEnd(head,30);
    printlist(head);
    break;
}
case 3:
{
    head=insertBegin(head,30);
    head=insertBegin(head,20);
    head=insertBegin(head,10);
    printlist(head);
    break;

}
case 4:
{
    Scanner sc = new Scanner(System.in);

    int data = 12, pos = 3;
    head = InsertPos(head, pos, data);
    System.out.print("Linked list after" + " insertion of 12 at position 3: ");
    printlist(head);


    // front of the linked list
    data = 1;
    pos = 1;
    head = InsertPos(head, pos, data);
    System.out.print("Linked list after" + "insertion of 1 at position 1: ");
    printlist(head);
    break;
}
case 5:
{
    head=delHead(head);
    printlist(head);
    break;
}
case 6:
{
    printlist(head);
    head= delLast(head);
    System.out.println("---------------");
    printlist(head);
    break;
}
case 7 :
{
    int index=2;
    head = dp(head, index);
```

```
        }
        case 8:
        {
            printlist(head);
            System.out.println("Position of element in Linked List: "+search(head,20));
            break;
        }
        case 9:
        {
            int c=count(head);
            System.out.println("number of elements "+c);
            break;
        }
        default :
        {
        System.out.println("exit");
        }

    }
}

public static void printlist(node1 head){
    node1 curr=head;
    while(curr!=null){
        System.out.print(curr.data+" ");
        curr=curr.next;
    }
}
```

```
static node1 insertEnd(node1 head, int x){
    node1 temp=new node1(x);
    if(head==null)
        return temp;
    node1 curr=head;
    while(curr.next!=null){
        curr=curr.next;
    }
    curr.next=temp;
    return head;
}

static node1 insertBegin(node1 head, int x){
    node1 temp=new node1(x);
    temp.next=head;
    return temp;
}

static node1 InsertPos(node1 headNode, int position, int data) {
    node1 head = headNode;
    if (position < 1)
        System.out.print("Invalid position");
    if (position == 1) {
        node1 newNode = new node1(data);
        newNode.next = headNode;
        head = newNode;
    } else
    {
        while (position-- != 0) {
            if (position == 1) {
                node1 newNode = new node1(data);
                newNode.next = headNode.next;
                headNode.next = newNode;
                break;
            }
            headNode = headNode.next;
        }
        if (position != 1)
            System.out.print("Position out of range");
    }
    return head;
}

static node1 delHead(node1 head){
    if(head==null)return null;// null null
    if(head.next==null){
        return null;//one node
    }
    else{
        node1 temp=head.next;
        // temp = head.next;
        temp.prev = null;
        head.next = null;
        //head.;//
        //head.prev.next=null;
        //head = head.next;
        return temp;
    }
```

```
}

static node1 delLast(node1 head){
    if(head==null)
        return null;
    if(head.next==null){
        return null;//single node list... null
    }
    node1 curr= head;
    while(curr.next.next!=null)
        curr = curr.next;
    //curr.prev.next = null;

    curr.prev = null;
    return head;
}

static int search(node1 head, int x)
{
    if(head==null)return -1;//corner cases
    if(head.data==x)return 1;//corner cases\\head
    else{
        int res=search(head.next,x);
        if(res==-1)return -1;
        else return res+1;
    }
}

public static int count(node1 head)
```

```
{
   node1 curr=head;
   int count = 0;
   while(curr!=null){
      System.out.print(curr.data+" ");
      count = count +1;
      curr=curr.next;
   }
   return count;
}

public static node1 dp(node1 head,int position)
{
   node1 headNode = head;
   node1 currNode = head;
   node1 nextNode = null;
   int index =0;
   if(position==0)
   {
      headNode = head.next;
      headNode.prev= null;
   }

   else
   {
   while(currNode.next!=null && index<position-1)
   {
      currNode = currNode.next;
      index++;
```

```
        }


        nextNode=currNode.next;

        currNode.next=nextNode.next;

        nextNode.next=null;

        }

        return headNode;

    }

}

class node1

{

    int data;

    node1 prev;

    node1 next;

    node1(int x)

    {

        data = x;

        prev = null;

        next = null;

    }

}
```