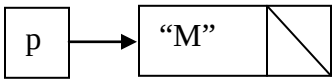
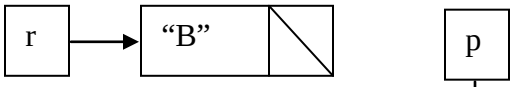
Question 1. (4 marks)
Trace the following code that uses `Node<String>` references.

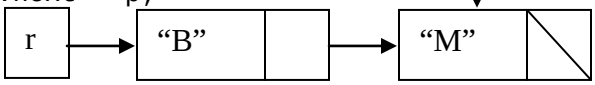
```
Node<String> p, q, r, s;

p = new Node<String>("M");
```



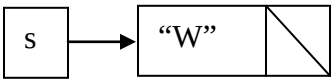

```
r = new Node<String>("B");
```



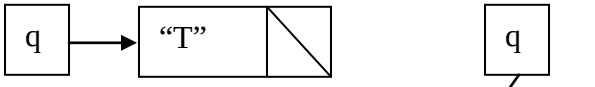

```
r.next = p;
```



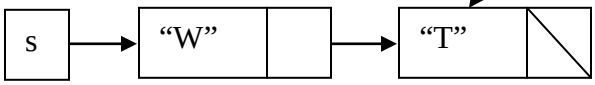

```
s = new Node<String>("W");
```



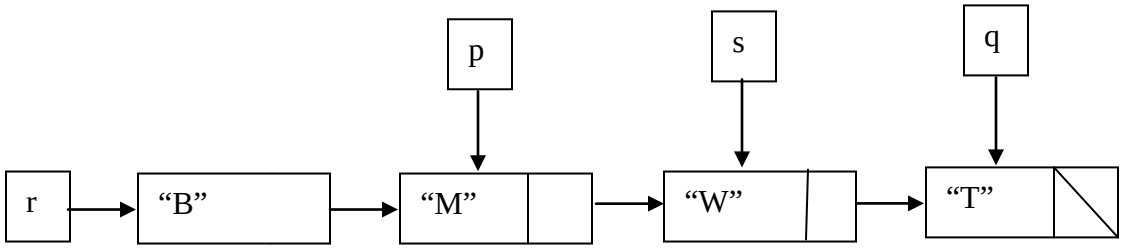

```
q = new Node<String>("T");
```



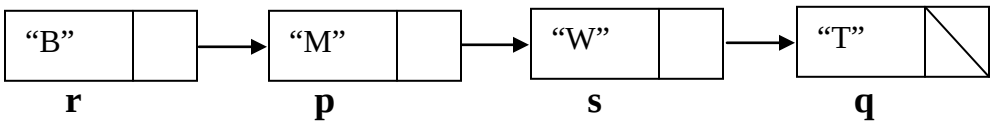

```
s.next = q;
```



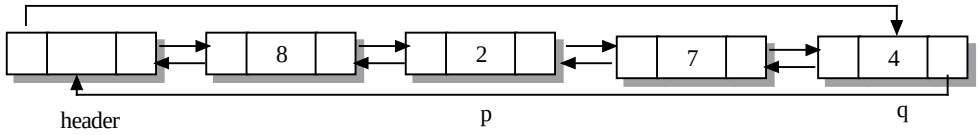

```
p.next = s;
```



After executing the above statements, show the order of the nodes inserted in the linked list by writing the node reference under each node and the value inside the node.



Question 2. (6 marks)
Use the following doubly linked list. Give the node value for the node referenced by each statement.



(a) `p.next = 7`

(b) `q.prev.prev = 2`

(c) `header.next.next = 2`

(d) `p.next.prev = 2`

(e) `q.next.next = 8`

(f) `header.prev = 4`

Question 3. (10 marks)

(a) Write a short code segment to traverse a linked list front-to-back and print out data in each node.

```
public static <T> void display(LinkList<T> l)
{
        boolean b = true;
        l.findfirst();

        while(b)
        {
                System.out.println(l.retrieve());

                if (l.last())
                        b = false;
                else
                        l.findnext();
        }
}
```

```
public void display()
{
        current = head;

        while(current != null)
        {
           System.out.println(current.data);
           current = current.next;
        }
}
```

```
public void display()
{
        findfirst();

        while(! last())
        {
            System.out.println(retrieve());
            findnext();
        }

        System.out.println(retrieve());
}
```

(b) Write a short code segment to traverse a linked list and print out data in every alternate node starting with the first node.

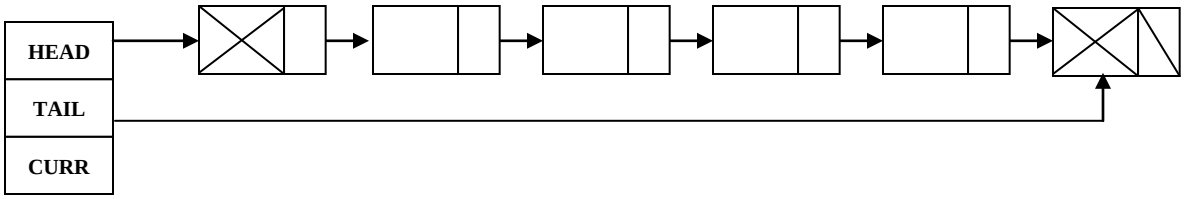
```
public static void displayAlt(LinkList<Integer> l)
{
        l.findfirst();
        int i = 1;

        while(! l.last())
        {
                System.out.println(l.retrieve());
                l.findnext();
                i++;

                if (! l.last())
                {
                        i++;
                        l.findnext();
                }
        }

        if (i % 2 != 0)
                System.out.println(l.retrieve());
   }
```

```
public void displayAlt()
{
        current = head;

        while(current != null)
        {
                System.out.println(current.data);
                current = current.next;

                if (current != null)
                        current = current.next;
        }
}
```

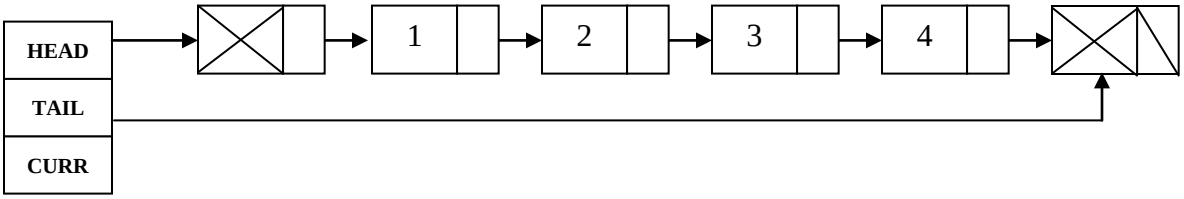
```
public void displayAltMethods()
{
        findfirst();
        int i = 1;

        while(! last())
        {
                System.out.println(retrieve());
                findnext();
                i++;

                if (! last())
                {
                        i++;
                        findnext();
                }
        }

        if (i % 2 != 0)
                System.out.println(retrieve());
   }
```

Question 4. (15 marks)

(a) Draw a graphical representation of an empty singly-linked list with sentinel nodes.



(b) Draw a graphical representation of a singly-linked list with sentinel nodes that has four elements (data) in it.



(c) Write class declaration of a doubly-linked list with sentinel nodes. Give only the data members and the constructor.

```
public class DNode <T>
{
      public T data;
      public DNode<T> next,prev;

      public DNode()
      {
            data = null;
            next = prev = null;
      }

      public DNode(T val)
      {
            data = val;
            next = prev = null;
      }
}
```

```
public class SentDLinkList <T>
{
    private DNode<T> head,tail;
    private DNode<T> current;

    public SentDLinkList()
    {
      head = current = new DNode<T>();
      tail = new DNode<T>();
      head.next = tail;
      tail.prev = head;
    }
}
```

Question 5. (12 marks)

(a) Write minimum number of statements to remove a node from the middle of a doubly-linked list.

```
current.next.prev = current.prev;
current.prev.next = current.next;
```

(b) Write minimum number of statements to insert a node in the middle of a doubly-linked list.

```
DNode<T> tmp = new DNode<T>(val);
tmp.next = current.next;
tmp.prev = current;
current.next.prev = tmp;
current.next = tmp;
```

Question 6. (8 marks)

What is the best- and worst-case time and space complexity of the operations **insert** and **remove** in:

(a) Array based list,

Insert best case is to insert at the end of the array O(1).
Insert worst case is to insert at the front of the array O(n).

Remove best case is to remove at the end of the array O(1).
Remove worst case is to remove at the front of the array O(n).

(b) Singly Linked list.

Insert best and worst case is the same O(1).

Remove best case is to remove at the front of the list O(1).
Remove worst case is to remove at the end of the list O(n).

(c) Doubly Linked list.

Insert and remove best and worst case is the same O(1).

(d) Singly Linked list with sentinel nodes.

Insert best and worst case is the same O(1).

Remove best case is to remove at the front of the list O(1).
Remove worst case is to remove at the end of the list O(n).