https://course.acciojob.com/idle?question=8507a3b7-ff9c-45c2-b12c-d886d2e145df

- MEDIUM
- Max Score: 40 Points
-

Reverse Linked List 2

Given the `head` node and two position values `m` and `n`. Your task is to reverse the linked list in the range of `m` amd `n`.

Input Format

The first line of contains 3 integers `N`, `m` and `n`. Where `N` is the number of nodes.

The second line of input contains `N` space separated integers denoting linked list.

Output Format

Your task is to return the head node after reversing the list.

Example 1

Input

```
6 2 4
1 2 3 4 5 6
```

Output

```
1 4 3 2 5 6
```

Explanation

```
The given linked list looks like 1->2->3->4->5->6->NULL , at second position we have 2 and at 4th position we have 4, so after reversing it becomes
1->4->3->2->5->6->NULL
```

Example 2

Input

```
1 1 1
5
```

Output:

```
5
```

Explanation:

```
Linked list looked like 5->NULL , so after reversing, the linked list remains same.
```

Constraints

1 <= N <= 500

-500 <= value of node <=500

1 <= left <= right <= N

Topic Tags

- **Linked lists**

My code

```
// n java
import java.io.*;
import java.util.*;
class Node
```

```
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }
class insertion
{
    Node head;
    Node tail;
        public void addToTheLast(Node node)
        {
          if (head == null)
          {
           head = node;
           tail = node;
          }
          else
          {
           tail.next = node;
           tail = node;
          }
        }
      void printList(Node head)
    {
        Node temp = head;
        while (temp != null)
        {
           System.out.print(temp.data+" ");
           temp = temp.next;
        }
```

```
        System.out.println();
    }
        /* Drier program to test above functions */

}
class Main
{
    public static void main(String args[])throws IOException
    {
         BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

            String S[] = br.readLine().split(" ");
                    int N = Integer.parseInt(S[0]);
            int m = Integer.parseInt(S[1]);
            int n = Integer.parseInt(S[2]);

                  String S1[] = br.readLine().split(" ");
                    insertion llist = new insertion();
                    int a1=Integer.parseInt(S1[0]);
                    Node head= new Node(a1);
            llist.addToTheLast(head);
            for (int i = 1; i < N; i++)
                    {
                          int a = Integer.parseInt(S1[i]);
                          llist.addToTheLast(new Node(a));
                    }

        Solution ob = new Solution();
              Node newhead=ob.reverseBetween(llist.head, m, n);
```

```
            llist.printList(newhead);
    }
}

class Solution
{
    public static Node reverseBetween(Node head, int m, int n)
    {
        //code here
            Stack<Node>sk=new Stack<>();
             Node p=head;
            Node ans=null;
            Node a=null;
            for(int i=1;i<m;i++)
                    {
                      p=p.next;
                    }
       for(int i=m;i<=n;i++)
                    {
                            sk.push(p);
                      p=p.next;
                    }
            if(m==1)
            {
                     ans=sk.pop();
                       a=ans;
                    while(!sk.empty())
                            {
                                    a.next=sk.pop();
                                    a=a.next;
```

```
                    }
                a.next=p;
                return ans;
            }
                ans=head;
                    a=ans;
        for(int i=2;i<m;i++)
                {
                a=a.next ;
                }
                        while(!sk.empty())
                    {
                        a.next=sk.pop();
                        a=a.next;
                    }
                a.next=p;
            return ans;
    }
}
```