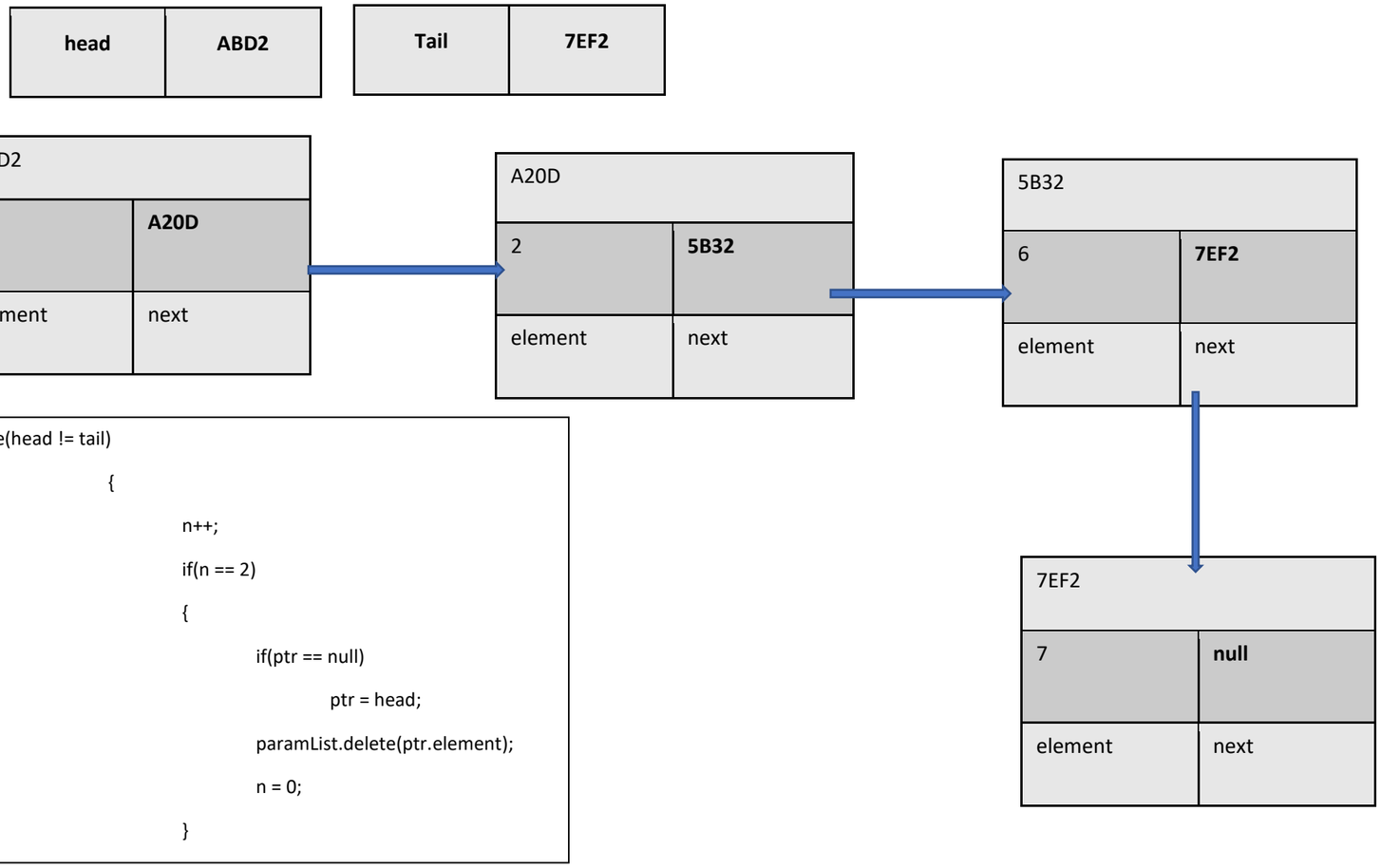
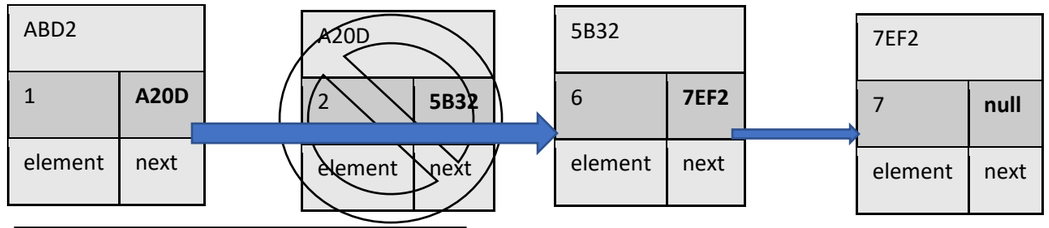
Morné Christo Stroebel 30155207

1. Design code

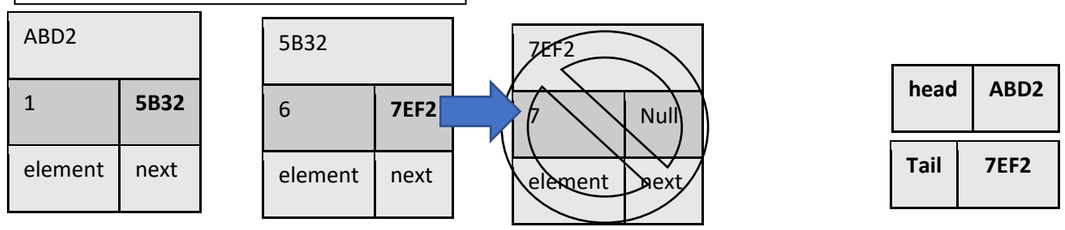



```
while(head != tail)
                {
                        n++;
                        if(n == 2)
                        {
                                if(ptr == null)
                                        ptr = head;
                                paramList.delete(ptr.element);
                                n = 0;
                        }
```

After one pass through josephus()



After two passes through josephus()

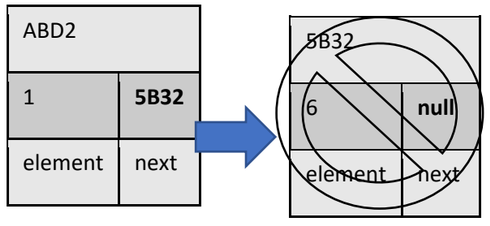


head	ABD2
Tail	7EF2

Here the pointer = null

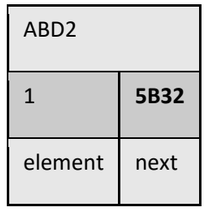
Pointer = head

After three passes through josephus()



head	ABD2
Tail	5B32

After four passes through josephus()



head	ABD2
Tail	ABD2

The while loop condition met

Return paramlist.tostring();

2. Code

a. Test class

```
public class prisoner implements Comparable<prisoner>
{
    private int id;
    private String firstname;
    private String lastname;

    public prisoner()
    {
        this(0, "", "");
    }

    public prisoner(int id, String firstname, String lastname)
    {
        setId(id);
        setFirstName(firstname);
        setLastName(lastname);
    }

    //set methods
    public void setId(int id)
    {
        this.id = id;
    }
    public void setFirstName(String firstname)
    {
        this.firstname = firstname;
    }
```

```
public void setLastName(String lastname)
{
	this.lastname = lastname;
}

//get methods
public int getId()
{
	return id;
}
public String getFirstName()
{
	return firstname;
}
public String getLastName()
{
	return lastname;
}

public int compareTo(prisoner g)
{
	String thisField = getFirstName();
	String otherField = g.getFirstName();
	return thisField.compareTo(otherField);
}
```

```
	@Override
	//toString method;
	public String toString()
	{
		return getId() + ". " + getFirstName() + " " + getLastName();
	}

	public static void main(String[] args)
	{
		prisoner s1 = new prisoner(1,"Janne","man");
		System.out.println(s1);
	}
}
```

b. New code for linked list

```
public String josephus(MyLinkedList <E> paramList)//My method to the josephus problem
{
    Node<E> ptr = head;
    int n = 0;
    if(head == null)//checks if the list is empty
        return "The list is empty!";
    while(head != tail
    {
        n++;
        if(n == 2)
        {
            if(ptr == null)
                ptr = head;
            paramList.delete(ptr.element);
            n = 0;
        }
        if(ptr == null)
        {
            ptr = head;
            n --;
        }
        else
            ptr = ptr.next;//get next element in stack
        }
        return "The survivor is: " + paramList.toString();
    }
```

c. Tests program

```
public class Test
{
    public static void main(String[] args)
    {
        StackAsMyLinkedList<Character> EmptyStack = new
        StackAsMyLinkedList<Character>();

        StackAsMyLinkedList<Integer> IntStack = new StackAsMyLinkedList<Integer>();

        StackAsMyLinkedList<String> StrStack = new StackAsMyLinkedList<String>();

        StackAsMyLinkedList<Character> CharStack = new
        StackAsMyLinkedList<Character>();

        StackAsMyLinkedList<prisoner> TestStack = new StackAsMyLinkedList<prisoner>();

        //Test data for variable type Integer!
        IntStack.push(new Integer(1));
        IntStack.push(new Integer(2));
        IntStack.push(new Integer(3));

        //Test data for variable type String!
        StrStack.push(new String("4. koos"));
        StrStack.push(new String("3. piet"));
        StrStack.push(new String("2. jan"));
        StrStack.push(new String("1. sannie"));

        //Test data for variable type Character!
        CharStack.push(new Character('a'));
        CharStack.push(new Character('b'));
        CharStack.push(new Character('c'));
```

```
//Test data for my test class prisoner!
TestStack.push(new prisoner(4,"koos","roos"));
TestStack.push(new prisoner(2,"Piet","pompies"));
TestStack.push(new prisoner(5,"jan","tuisbly"));
TestStack.push(new prisoner(99,"pieter","boos"));
TestStack.push(new prisoner(189,"Janco","hoog"));
/*
    The Josephus problem is as follow X number of people stands in a circle
    waiting to be executed by the person next to him, except one person

    will survive. So, my method will calculate witch person in the group stands in
    the surviving position!
*/
System.out.println("The Josepuhs problem!");
System.out.println("-------------------------------------");
//The test with empty list
System.out.println("\nEmpty list:");
System.out.println("-------------------------------------");
System.out.println(EmptyStack);
System.out.println(EmptyStack.josephus());
```

```
        //Test with one item:
        EmptyStack.push(new Character('a'));
        System.out.println("\nList with one valaue");
        System.out.println("-------------------------------------");
        System.out.println(EmptyStack);
        System.out.println(EmptyStack.josephus());
        //The test with Integer valaues
        System.out.println("\nInteger:");
        System.out.println("-------------------------------------");
        System.out.println(IntStack);
        System.out.println(IntStack.josephus());
        //The test with String valaues
        System.out.println("\nString:");
        System.out.println("-------------------------------------");
        System.out.println(StrStack);
        System.out.println(StrStack.josephus());
        //The test with Character valaue
        System.out.println("\nCharacter:");
        System.out.println("-------------------------------------");
        System.out.println(CharStack);
        System.out.println(CharStack.josephus());
        //The test with testclass valaue
        System.out.println("\nTest class prisoner:");
        System.out.println("-------------------------------------");
        System.out.println(TestStack);
        System.out.println(TestStack.josephus());
    }
}
```

3. Screenshot of output

```
Empty list:
-------------------------------------

The list is empty!

List wit one valaue
-------------------------------------
a
The survivor is: a

Integer:
-------------------------------------
3
2
1
The survivor is: 1

String:
-------------------------------------
1. sannie
2. jan
3. piet
4. koos
The survivor is: 1. sannie

Character:
-------------------------------------
c
b
a
The survivor is: a

Test class prisoner:
-------------------------------------
189. Janco hoog
99. pieter boos
5. jan tuisbly
2. Piet pompies
4. koos roos
The survivor is: 5. jan tuisbly
```

4. Analysis with simplified model

```
53      public String josephus(MyLinkedList <E> paramList)//My metl
54      {
55          Node<E> ptr = head;
56          int n = 0;
57          if(head == null)//checks if the list is empty
58              return "The list is empty!";
59          while(head != tail)//this while loop will continue whi
60          {
61              n++;
62              if(n == 2)//every time n reach aa valaue of 2 the p
63              {
64                  if(ptr == null)//since the pointer valaue runs
65                      ptr = head;
66                  paramList.delete(ptr.element);
67                  n = 0;//resets n to 0
68              }
69              if(ptr == null)//ptr needs to start again at the h
70              {
71                  ptr = head;
72                  n --;
73              }
74              else
75                  ptr = ptr.next;//get next element in stack
76          }
77          return "The survivor is: " + paramList.toString();
78      }
```

55. Tfetch + Tstore

56. Tfetch + Tstore

57. Tfetch + T<

58. Tfetch + Treturn

59. (2Tfetch + T<)(n)

61. 2Tfetch + T+ + Tstore

62. 2Tfetch + T<

64. Tfetch + T<

65. 2Tfetch + Tstore

66. Tfetch + T[.]

67. Tfetch + Tstore

69. Tfetch + Tstore

71. 2Tfetch + Tstore

72. Tfetch + T-

74. 2Tfetcch + T+ + Tstore

77. Tfetch + Treturn

= 3(n) + 1+1+1+1+1+1+1+1+2+1+1+2+1+1+1+2+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1

= 3n + 35