פתרון מוצע קיימים עוד פתרונות שונים אבל אלו הפתרונות המוצעים על ידי פתרונות שונים קיבלו ניקוד מלא בהתניה שלא חרגו מהסינטקס תקין וקיבלו ניקוד חלקי על כיוון נכון

1)א

```
public static boolean func1(node<String> chain)
{
    node<String> temp =chain;
    int index = 0;
    while(temp!= null)
    {
        int need_to_be_0 =index%2+temp.getValue().length()%2;
        if(need_to_be_0!=0)
            return false;
        ++index;
        temp = temp.getNext();
    }
    return true;
}
```

1)ב

```
public static boolean func1(node<Integer> chain)
{
    node<Integer> temp =chain;
    while(temp!= null)
    {
        if(!check(temp))
            return false;
        temp =temp.getNext();
    }
    return true;
}
public static boolean check(node<Integer> temp)
{
    int val = temp.getValue();
    temp =temp.getNext();
    boolean flag = false;
    while(temp!=null)
    {
        if(temp.getValue()==val)
        {
            return true;
        }
        temp =temp.getNext();
    }
    return false;
}
```

(2א)

```
public static boolean xCy(String str)
{
    boolean flag = true;
    if(str.length()%2==0||!str.contains("C"))
        return false;
    Stack<Character> first_half = new Stack<Character>();
    Queue<Character> second = new LinkedList<>();
    /// split the string char by char
    for (int i = 0; i < str.length(); i++)
    {
        if(str.charAt(i)=='C')
        {
            flag =false;
            continue;
        }
        if(flag)
        {
            first_half.push(str.charAt(i));
        }
        else
        {
            second.add(str.charAt(i));
        }
    }
    /// check values
    if(first_half.size()!=second.size())
            return false;
    while(!first_half.isEmpty())
    {
        if(first_half.pop()!=second.poll())
        {
            return false;
        }
    }
    return true;
}
```

(2ב)א)

```
public void q2a(Stack<Integer> s)
{
    Stack<Integer> temp = new Stack<>();
    while(!s.isEmpty())
    {
        temp.push(s.pop());
    }
    while(!temp.isEmpty())
    {
        System.out.println(temp.peek());
        s.push(temp.pop());
    }
}
```

2)ב)ב

```java
public void q2b(Stack<Integer> s)
{
    Stack<Integer> temp = new Stack<>();
    while(!s.isEmpty())
    {
        System.out.print(s.peek()+",");
        temp.push(s.pop());
    }
    while(!temp.isEmpty())
    {
        s.push(temp.pop());
    }
}
```

3)

```java
public  LinkedList<int[]> q3(node<Integer> chain)
{
    LinkedList<int[]> answer = new LinkedList<>();
    while(chain!=null)
    {
        int limit = chain.getValue();
        int [] ans = new int[limit+1];
        ans[limit] = 0;
        for (int i = 0; i < limit; i++)
        {
            ans[i]=(int)Math.pow(2,i);
        }
        answer.add(ans);
        chain = chain.getNext();
    }
    return answer;
}
```

4) ניתן פשוט להריץ את הקוד בקומפיילר שלכם ולראות שהפלט בשני המקרים הוא

YAA its cool
safe side
it's a boy

(5

```
public static boolean isSymetric(Queue<Integer> q1,Queue<Integer> q2)
{
    if(q1.size()!=q2.size())
        return false;
    boolean ans = true;
    Stack<Integer>s1 = new Stack<>();
    while(!q1.isEmpty())
    {
        if((int)q1.peek() != (int)q2.peek())
        {
            ans = false;
            break;
        }
        s1.push(q1.poll());
        q2.add(q2.poll());
    }
    while(!s1.isEmpty())
    {
        q1.add(s1.pop());
    }
    return ans;
}
```

6)

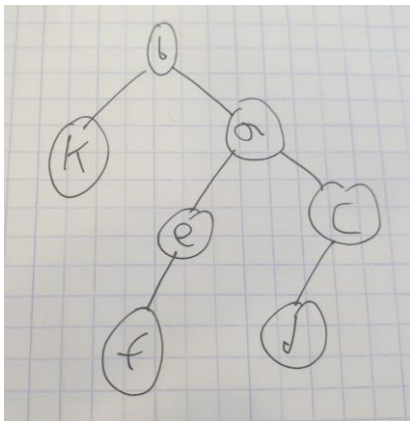
```
// Function to search and delete all occurrences of value from tree.
BinNode Delete(BinNode root, int data) {
if(root == null) return root;

if(root.data == data)
{
    // Case 1:  No child
    if (root.getLeftChild() == null && root.getRightChild() == null)
        root = null;
    //Case 2: One child
    else if (root.getLeftChild() == null)
    {
        root = root.getRightChild();
    } else if (root.getRightChild() == null)
    {
        root = root.getLeftChild();
    }
    // case 3: 2 children
    else {
        BinNode temp = find_min(root.getRightChild());
        root.setData(temp.getData());
        temp = null;
    }
}
    if(root == null) return root;
    root.setLeftChild(Delete(root.getLeftChild(),data));
    root.setRightChild(Delete(root.getRightChild(),data));
    return root;
}

private BinNode find_min(BinNode rightChild)
{
    while(rightChild.getLeftChild()!=null)
    {
        rightChild=rightChild.getLeftChild();
    }
    return rightChild;
}
```

7) א)

הפלט יהיה:

a c c e e f f f e c a b d d d b b a

7)ב)



8)א)

```
public static node<Integer> Q8(node<Integer> chain)
{
    LinkedList<Integer> appeared = new LinkedList<>();

    node<Integer> ans = new node<>(chain.getValue());
    node<Integer> ans_point = ans;

    appeared.add(chain.getValue());
    node<Integer> temp =chain;
    temp = temp.getNext();
    while(temp!= null)
    {
        if(!appeared.contains(temp.getValue()))
        {
            appeared.add(temp.getValue());
            ans_point.setNext(new node<>(temp.getValue()));
            ans_point = ans_point.getNext();
        }
        temp = temp.getNext();
    }
    return ans;
}
```

8)ב)

```
public static int Q8b(String str)
{
    LinkedList<Integer> int_vals = new LinkedList<>();
    LinkedList<Character> char_vals = new LinkedList<>();
    for (int i = 0; i < str.length(); i++) {
        char cur = str.charAt(i);
        if(cur>='0'&&cur<='9')
```

```
            int_vals.add(cur-'0');
        if(cur>='a'&&cur<='z')
            char_vals.add(cur);
        if(cur>='A'&&cur<='Z')
            char_vals.add(cur);
    }
    return  char_vals.size()+int_vals.size();
}
```

9)א)

```
public static void Q9a(CellPtr cell)
{
    if(cell == null)
        return;
    System.out.println(cell.getValue());
    Q9a(cell.getNext());
}
```

9)ב)

```
public static void Q9b(CellPtr cell)
{
    if(cell == null)
        return;
    System.out.println(cell.getValue());
    Q9b(cell.getPrev());
}
```

9)ג)

```
public static CellPtr Q9c(CellPtr cell, int input)
{
    CellPtr new_cell = new CellPtr(input);
    cell.setNext(cell);
    return new_cell;
}
```

(10

```
public static  node<Integer> q10(node<Integer> chain)
{
    while(chain!=null&&chain.getValue()==0)
    {
        chain = chain.getNext();
    }
    if(chain == null) return null;
    node<Integer> ans = new node<>(chain.getValue());
    node<Integer> ans_pointer = ans;
    while(chain!=null)
    {
        int limit = chain.getValue();
        for (int i = 0; i < limit; i++) {
            ans_pointer.setNext(new node<Integer>(limit));
            ans_pointer = ans_pointer.getNext();
        }
        chain = chain.getNext();
    }
    return ans;
}
```

(11

```
class Person
{
    String name;
    int age;
    String id;
    boolean sex;

    public Person(String name, int age, String id, boolean sex) {
        this.name = name;
        this.age = age;
        this.id = id;
        this.sex = sex;
    }

    public void print()
    {
        System.out.print("name:"+this.name);
        System.out.print("age:"+this.age);
        System.out.print("id:"+this.id);
        if(sex)
            System.out.print("sex:male");
        else
            System.out.print("sex:female");
    }
}
```

```
class Worker extends Person
{
    int salary;
    public Worker(String name, int age, String id, boolean sex) {
        super(name, age, id, sex);
```

```
    }
    public void print()
    {
        super.print();
        System.out.println("salary:"+this.salary);
    }
}
```

```
class Student extends Person
{
    String school;
    String subject;


    public Student(String name, int age, String id, boolean sex) {
        super(name, age, id, sex);
    }

    public void print()
    {
        super.print();
        System.out.println("learns "+this.subject+" at
"+this.school);
    }
}
class Teacher extends Person
{
    String subject;
    int exp;


    public Teacher(String name, int age, String id, boolean sex) {
        super(name, age, id, sex);
    }

    public void print()
    {
        super.print();
        System.out.println("teach "+this.subject);
        System.out.println("teach "+this.exp+"years already");
    }
}
```

```
public static void main(String[] args) {
    ///1
    Cinema c = new Cinema("cinema city",50,15,7);
    ///2
    Person[] arr =new Person[15];
    //3
    for (int i = 0; i < arr.length; i++) {
        def_person(arr[i]);
    }
    ///4
    price_cal(arr,c);
}
```

```java
public static void def_person(Person p)
{
    Random rand = new Random();
    int num = rand.nextInt(3);
    switch (num) {
        case 0:
            p = new Student("aaa",18,"1111111111",true);
        case 1:
            p = new Teacher("bbb",50,"000000000",false);
        case 2:
            p = new Worker("ccc",24,"222222222",true);
    }
}
```

```java
public static void price_cal(Person[] p,Cinema c)
{
    double sum = 0;
    int stud_num = 0;
    int teacher_num = 0;
    int worker_num = 0;
    for (int i = 0; i < p.length; i++) {
        if(p[i] instanceof Student)
        {
            sum+=c.price*c.student_discount/100;
            stud_num++;
        }
        else if(p[i] instanceof Teacher)
        {
            sum+=c.price*c.teacher_discount/100;
            teacher_num++;
        }
        else
        {
            sum+=c.price;
            worker_num++;
        }
    }
    System.out.println("total income ="+sum);
    System.out.println("movie price ="+c.price);
    System.out.println("total watchers ="+p.length);
    System.out.println("student number ="+stud_num);
    System.out.println("teacher number ="+teacher_num);
    System.out.println("worker number ="+worker_num);
}
```