1. Create a class Player with the following details

Data members : Name, PlayerID, Round1Score, Round2Score

-Create a Linked list with objects of Player class, Iterate through the list and remove players whose avg score is <60.

-print the final list of qualified players

-bonus task : Try to find a way to print the object directly.
Like this: System.out.println(obj);

```
import java.util.LinkedList;
public class Player {

    static LinkedList<Player> players=new
LinkedList<Player>();
    String name;
    int playerID,round1Score, round2Score;


    Player(int id,String name,int m1,int m2){
        playerID=id;
        this.name=name;
        round1Score=m1;
        round2Score=m2;
    }

    @Override
    public String toString() {
        return "name: "+name;
    }
```

```
    public static void main(String[] args) {

        players.add(new Player(3,"Aman",65,85));
        players.add(new Player(6,"Syed",80,50));
        players.add(new Player(8,"Anu",73,78));
        players.add(new Player(4,"Riya",48,50));
        players.add(new Player(1,"Raj",75,44));

        for (Player p:players) {
            if((p.round1Score+p.round2Score)/2<60){
                players.remove(p);
            }
        }
        System.out.println("Qualified players :");
        for(Player p : players){
            System.out.println(p);
        }
    }
}
```

2. Write a program to input a sentence from a user
- create a hashmap with all the characters in the sentance as key, and their frequency in the sentance as value

- Print the map
-print all the character which has the highest occurence

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class KeyCount {
```

```
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        Map<Character,Integer> map = new
HashMap<>();
        System.out.println("Enter a sentence :");
        char[] chArray
=sc.nextLine().toLowerCase().toCharArray();
        for(char ch : chArray){
            if(Character.isLetterOrDigit(ch)) {
                if (map.containsKey(ch))
                    map.replace(ch, map.get(ch) + 1);
                else
                    map.put(ch, 1);
            }
        }
        System.out.println("Map : \n"+map);

        int big =0;
        for(int val : map.values()){
            if(val>big)
                big=val;
        }
        System.out.println("Highest frequency : "+ big);
        System.out.print("Characters : ");
        for (char key : map.keySet()) {
            if(map.get(key)==big)
                System.out.print(key +" ");
        }
    }
}
```

3. Create a class square with following details

-Input a 5+ digit number from the user

-method primeFactorise() to prime factorise the number and to store the frequency of each factor in a map

-method isPerfectSquare() to check if the number is a perfect square by using the map

Print the map and whether the number is a square or not.
Hint: number is a square if frequency of each factor is even.

Eg : 122500
Map : 2=2, 5=4, 7=2
The number is a perfect square

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Square {
    int n;
    Map<Integer,Integer> map=new
HashMap<Integer,Integer>();

    boolean isPerfectSquare(){
        for (int a : map.values()) {
            if(a%2!=0)
                return false;
        }

        return true;
    }

    void primeFactorise(){
        int pf=2; //prime factor
```

```
        int num = n,value;
        while(num>1){
            if(num%pf==0) {
                map.put(pf, map.getOrDefault(pf,0)+1);
//Alternate method
//                if (map.containsKey(pf)) {
//                    value = map.get(pf);
//                    map.replace(pf, value + 1);
//                }
//                else
//                    map.put(pf,1);
                num/=pf;
            }
            else
                pf++;
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Square s = new Square();
        System.out.println("Enter the number :");
        s.n=sc.nextInt();
        s.primeFactorise();
        System.out.println("Prime factorization :\
n"+s.map);
        if (s.isPerfectSquare())
            System.out.println("The number is a perfect
square");
        else
            System.out.println("Number is not a perfect
square");
    }
}
```

4. Input a Bracket expression from the user and check if it is balanced using a stack.

A bracket expression consists of the following charecters
{ } ( ) [ ] < >
An expression is balanced if :
->Every opened bracket is closed by the same kind of bracket
-> brackets opened last are closed first

Eg : { [() <{}>]()} : balanced
{ <}> : not balanced

Logic Worksheet

1. Sumproduct number : sumOfDigits * productOfDigits = num
144 -> (1+4+4)*1*4*4 = 144

2. Magic number : eventual sum of digits is 1
289-> 2+8+9=19 -> 1+9=10 -> 1+0 = 1

3. Spy number : sum pf digits = product of digits
eg 1111, 2222

Solution will be shared later.