1. (5 pts)

Write a (single) for loop that prints the following 7 lines of text:

yes
no
yes
no
yes
no
yes

for(int j = 0; j < 7; j++){
    if (j%2==0) System.out.println("yes"); else System.out.println("NO");
}

2. (5 pts)

Write a (single) for loop that prints the integers from -300 to 300 in intervals of 50:

-300
-250
-200
-150
-100
-50
0
50
100
150
200
250
300
for(int j = -300; j <= 300; j = j + 50){
    System.out.println(j);
}

3. (5 pts)

Suppose s is any String. Write a code fragment that prints the characters in s in a column with every other character doubled. You must use a `for` loop in your solution.

For example if s = "Donkey", your code should print

```
D
oo
n
kk
e
YY
```

If s = "snake", your code should print

```
s
nn
a
kk
e
```

for(int j = 0; j < s.length(); j++){
    if (j%2==0) System.out.println(s.charAt(j);
    else{ System.out.print(s.charAt(j)); System.out.println(s.charAt(j);}
}

4. (5 pts)

Write a `for` loop that prints all integer values from 10 down to -5 except 7 and -1. Thus your output should look like this:

```
10
9
8
```
5. (5 pts)

Suppose you are given an integer variable N (of type int), and suppose N > 0. Write a for loop that sums the reciprocals of the positive integers less than or equal to N.

The reciprocal of a number is the value 1.0 divided by that number. For example the reciprocal of 2 is 1.0/2, or 0.5.

If N = 5 your loop should add up these fractions:

1.0/1 + 1.0/2 + 1.0/3 + 1.0/4 + 1.0/5

which is approximately 2.28.

double sum = 0.0;
for(int j = 1; j <= N; j++) sum += 1.0/j;

6. (25 pts)

Write a complete, working program in a single class called ExactCount.

Here is what ExactCount should do: first, it should read in a String,
say \( s \), and an \( \text{int} \), say \( k \), from the keyboard. Then it should check if the symbol '$' appears in \( s \) exactly \( k \) times. If it appears exactly \( k \) times, your code should print the word "YES"; otherwise your code should print the word "NO".

Use an object from the \texttt{Scanner} class to read in the String and int values.

Here are two examples of program behavior:

Enter test string:
\[
\text{[user types value below]}
\]
Bus$$Drive$r
Enter integer value:
\[
\text{[user types value below]}
\]
5
NO

---

Enter test string:
\[
\text{[user types value below]}
\]
My dog $\text{likes}$ cats!
Enter integer value:
\[
\text{[user types value below]}
\]
2
YES

```java
import java.util.Scanner;
public class ExactCount{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        int ct = sc.nextInt();
        int n = 0;
        for(int j = 0;j<s.length();j++){
            if(s.charAt(j) == '$') n++;
        }
        if(ct == n) System.out.println("YES");
        else System.out.println("NO");
    }
}
```

7. (25 pts)

The \texttt{FuelTruckTester} code, below, makes use of a class called
FuelTruck; the FuelTruck class is intended to model the operation of a gasoline truck.

```java
public class FuelTruckTester{
    public static void main(String[] args){
        //gasoline truck: (String) driver, (int) truck capacity, (int) amount in truck
        FuelTruck t = new FuelTruck("Dave",10000,8500);
        //assign new driver to t
        t.setDriver("Dana");
        //amount in truck
        System.out.println("Load amount: " + t.amount());
        //value of current amount, that is, the number of gallons, at $3.91/gallon
        System.out.printf("Load value: $%8.2f%n", t.retailValue(3.91));
        //unload 3000 gallons..
        t.unload(3000);
        System.out.println("New load amount: " + t.amount());
        System.out.printf("new value: $%8.2f%n", t.retailValue(3.91));
    }
}
```

Here is the output from the FuelTruckTester run above:

Load amount: 8500
Load value: $33235.00
New load amount: 5500
new value: $21505.00

The FuelTruck class does not exist. Your job here: create the FuelTruck class so that the FuelTruckTester code runs correctly.

```java
public class FuelTruck{
    private String driver;
    private int capacity;
    private int amount;

    public FuelTruck(String d, int c, int a){
```
driver = d;
capacity = c;
amount = a;
}

public void setDriver(String w){driver = w;}
public String getDriver(){return driver;}
public int amount(){return amount;}
public double retailValue(double c){
  return(c*amount);
}
}

8. (5 pts)

You are working on software to support the UMass hiking club. You are preparing a Hike class that will help hikers choose a hike. A Hike object describes a hike by name, along with the hike distance and the expected time the hike takes.

The partially implemented Hike class is given below:

public class Hike{

  private String name;
  private double distance; // in miles
  private double time; // in hours

  public Hike(String nm, double dist, double time){
    // constructor implementation goes here
  }

  public String getName(){return name;}
  public double getDistance(){return distance;}
  public double getTime(){return time;}

}
For this question complete the implementation of the Hike constructor. Do not alter the constructor header line; simply enter the constructor body in the box below.

```java
name = nm;
distance = dist;
this.time = time;
```

9. (5 pts)

The Hike class is above. The HikingDriver class is given below. Notice that it has an empty main method.

```java
public class HikingDriver{
    public static void main(String[] args){
    }
}
```

For this question add two statements to the HikingDriver class. The first should construct a Hike object called `niceHike`. This Hike object should have name field "BlackLake", distance 5.1 miles, and time 2.0 hours. The second statement to add should use a method from the Hike class to print the length (the distance value) of niceHike to the console.

```java
Hike niceHike = new Hike("BlackLake",5.1,2.0);
System.out.println(niceHike.getDistance());
```

10. (5 pts)

The Hike class is above. For this question add a new method to the Hike class called rename; this method should take one parameter, the new name of the hike, and should change the name field of the calling object to that new name. For example, if `niceHike` has name "Black Lake", then this statement:

```java
niceHike.rename("Blue Lake");
```
should change the name field of `niceHike` to "Blue Lake".

```java
public void rename(String n){name = n;}
```

11. (5 pts)
The `Hike` class is above. For this question add a new method to the `Hike` class called `milesPerHour`; this method should take no parameters, and should return the miles per hour the calling `Hike` object is supposed to take. For example if `sallysHike` is a hike that is 6.0 miles long and takes 3.0 hours, then this statement:
```java
double mph = sallysHike.milesPerHour();
```

should assign to the variable `mph` the value 2.0 (the distance of the hike divided by the number hours the hike takes).

```java
public double milesPerHour(){return distance/time;}
```

12. (5 pts)
The `Hike` class is above. For this question add a new method to the `Hike` class called `quickHike`; this method should take no parameters, and should return `true` if the calling `Hike` object takes strictly less than 1.0 hours; otherwise the method should return `false`. For example if `someHike` is a hike that is 6.0 miles long and takes 3.0 hours, then this statement:
```java
boolean fast = someHike.quickHike();
```

should assign to the variable `fast` the value `false`, since `someHike` takes more than one hour.

```java
public boolean quickHike(){return (time < 1.0);}