Q1. Write a program which starts two threads marked “even” and “odd”. The threads cooperate to print the numbers from 1 to 20 in sequence with the “even” thread printing only even numbers and odd the thread printing only odd numbers.
Ignore output line order.

Output Format

The numbers from 1 to 20 are printed in sequence alternatively by the odd and even threads with the even thread printing only even numbers and the odd thread printing only odd numbers.

Sample Input

```
```

Sample Output

```
Thread Odd: 1
Thread even: 2
Thread Odd: 3
```

Time Limit: 100 ms Memory Limit: 256 kb Code Size: 1024 kb

Q2. Maruti Suzuki gives a special bonus to its employees with 100% attendance in a year with a certificate of appreciation. Assume there are N employees. Get the attendance percentage of N employees and store them in an array.

Create **two threads** so that thread1 determines the total count of employees eligible for a certificate in the first half of the array and thread2 in the second half of the array.

The main( ) has to wait till both the threads complete their task and arrive at a final count indicating the total number of employees eligible for the certificate of appreciation.

Input Format

The first line of the input consists of the number of employees.
The second line has the average attendance percentage of employees.

Output Format

The output prints the winners count.

Sample Input

```
10
89 100 75 98 90 100 100 98 83 99
```

Sample Output

```
Winners : 3
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb Q3. **Stall Revenue**

The Accounting department of the fair committee wants an console application that can estimate the total revenue by rent from an exhibition. So write a program that accepts the stall details of an exhibition that includes the stallArea which is used for computing the stallCost. Using threads, calculate the stallCost of each stalls and in the main method, print the consolidated data.

Create a class **Stall** which implements Runnable interface with the following private attributes,

Create default constructor and a parameterized constructor with arguments in order Stall(String stallName, String details, Double stallArea, String owner). Create appropriate getters and setters.
Override the following methods in the **Stall** class,

Get the number of stalls and stall details and calculate the total revenue of all the stalls. Calculate the stall cost for each stall, each cost will be calculated by seperate thread.
Create a driver class **Main** to test the above requirements.

Input Format

First line of the input consist of the number of inputs to be given Next input is the stall details
Refer sample input

Output Format

Output prints the stall cost of each stall Refer sample output

Sample Input

```
3
Book stall
Stall for books
```

Sample Output

```
3750.0
4500.0
9000.0
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q4. **Profit or Loss**

Now we are going to create a console application that can estimate whether the booking is a profit or loss, thereby enabling hall owners to reduce or increase expenses depending on the status. Hence if the several Booking details are given, compute whether the bookings are profitable or not. Use Threads to compute for each booking, Finally display the profit/loss status.
Create a class HallBooking which implements Runnable interface with following private attributes.

Include appropriate getters and setters.
Create default constructor and a parameterized constructor with arguments in order HallBooking(String hallName, Double cost, Integer hallCapacity,Integer seatsBooked).
Override run() method which display the status (i.e) Profit or Loss.If SeatsBooked*100 > cost then it is a profit else loss .
Create a driver class Main. The status for each hall is calculated by separate threads. The Threads print the status of the events.
Ignore output line order

Input Format

The first line of input corresponds to the number of events 'n'. Next input is the hall details.
Refer sample input for formatting specifications.

Output Format

The output consists of the status of the events. (Profit or Loss). Refer sample output for formatting specifications.

Sample Input

```
4
Le Meridian 12000
```

Sample Output

```
Profit Loss Profit
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q5. Write a program for matrix multiplication in java using threads.

Input Format

Row and column value of Matrix 1 in first line separated by space Row and column value of Matrix 2 in second line separated by space Matrix 1 elements
Matrix 2 elements

Output Format

Display the matrix after multiplication.

Constraints

Integers only.

Sample Input

Sample Output

```
2 2
2 2
1 2
```

```
19 22
43 50
```

Time Limit: 10 ms Memory Limit: 256 kb Code Size: 1024 kb

Q6. **Interest Calculation**

Now we are going to calculate the interest based on the balance for a bank application. Use threads to calculate the interest and final amount. Finally print the interest and the final balance.

Create a class **Account** that extends thread class with the following private attributes.

Include appropriate getters and setters.
Create default constructor and a parameterized constructor.
Override the run() method to calculate and display the interest and balance.
If the balance is greater than or equal to 10000 then the rate of interest is 8% else 5%.
Create a driver class **Main.** The interest and balance for each account is calculated by separate threads. The Threads print the interest and final balance of the accounts.
Ignore Output line order.

Input Format

The first line of the input consists of the input n. Next input is the account details.

Output Format

The output prints the interest and balance of the accounts in next lines.

Sample Input

```
2
3256858548
50000
```

Sample Output

```
250.00
4000.00
5250.00
```

Time Limit: - ms Memory Limit: - kb Code

Size: - kb Q7. **ItemType-Amount**

Calculation

Now we are going to calculate the total amount of the items purchased based on the number of items and cost per item. Use threads to calculate the total amount.
Create a class **ItemType** that extends thread class with the following private attributes.

Include appropriate getters and setters.
Create default constructor and a parameterized constructor.
Override the run() method to calculate and display the total amount. Round off the double values to two decimal places.
Create a driver class **Main.** The total amount is calculated by separate threads. The Threads print the total amount of the items purchased.

Input Format

First line of the input consist of number of inputs to be given Next input is the item details
Refer sample input

Output Format

Output prints the total amount of the items purchased. Refer sample output

Sample Input

```
2
Laptop
40000
```

Sample Output

```
300000.00
400000.00
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb

Q8. Your English literature friend is very happy with the code you gave him. Now for his research, he used your application to find

character frequency in many novels. For larger novels, the application takes a lot of time for computation. So he called you on a fine Sunday to discuss this with you. He wanted to know whether you can improve the speed of the application.

You decided to modify the application by using multiple threads to reduce the computation time. For this, accept the number of counters or threads at the beginning of the problem and get the string for each counter or thread. Create a thread by extending the Thread class and take the user entered the string as input. Each thread calculates the

character frequency for the word assigned to that thread. All the counts are stored locally in the thread and once all the threads are completed print the character frequency for each of the thread.

Ignore output line order.

Input Format

Number of String (N)
N number of Strings in each line

Output Format

Frequency of characters as shown in sample output

Sample Input

```
2
welcome java
```

Sample Output

```
w1 e2
```

Time Limit: - ms Memory Limit: - kb Code Size: - kb