SkillsUSA Programming - State

An alien tourism company specializing in touring the cosmos had a special Earth vacation package. Many aliens across many planets couldn't let this opportunity pass, so they found themselves enjoying themselves here on Earth. Unfortunately for them, they collided with a satellite while entering the atmosphere and fried their mainframe system. To make matters worse, the computer components on Earth are incompatible with their systems, so they have to wait until the next ship arrives before they can leave.

Their jewelry business was a success, thanks to your help finishing their programs. However, as they are still stranded on Earth, they have decided to change focus. Some Aliens are working on studying human-made satellites on Earth, as the aliens did not utilize satellites for their services on their home planets. Others have decided to get into video game development, working on the highly anticipated game *DigBuild*.

They are once again asking for your coding support.

Instructions

You must download the zip file for your language, and using the instructions sent last week, load it into your IDE or editor of choice.

You are to only edit `AlienHelp.java` or `AlienHelp.cpp`, as the rest of the code is given to help you.

Problem 1: Orbit

The aliens who are studying satellites have a fascination with geostationary orbit. They want to create a program that, given the radius of the planet and the desired altitude of the satellite, will output the necessary velocity of the satellite to maintain geostationary orbit. Geostationary orbit is when a satellite is in the same location relative to the location on the surface of the planet. See diagram:

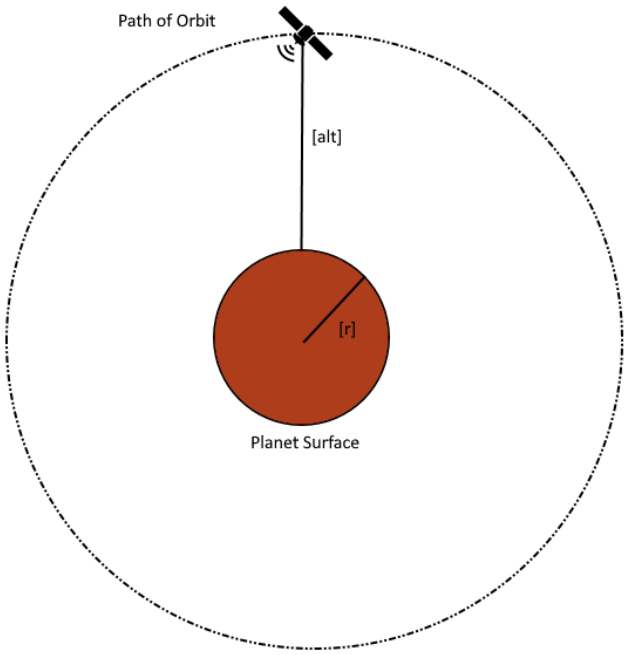

Figure 1: orbit-diagram.png

However, they skipped the introductory math course and programming course and thus can't figure out how to calculate it.

Given a value of `radius` and `altitude`, you must calculate the speed of orbit.

You may assume `radius` and `altitude` is in kilometers. Please return your answer in Kilometers per Hour, rounded to the nearest whole number.

Assume `pi` is 3.14.

Assume there are 24 hours in a day, 60 minutes in an hour, and 60 seconds in a minute exactly.

Please do not use constants provided in the programming language or more precise measurements.

You must return this calculated value **ROUNDED TO THE NEAREST 3 DECIMAL PLACES**.

Problem 1: Function Arguments

```
parameter: radius: Radius of the planet (distance from the core to the surface)
parameter: altitude: Altitude from the surface of the planet at which the satellite is orbiting
```

Problem 1: Return Value

```
return: int: The speed at which the satellite must travel at the altitude to maintain geostationary orbi
```

Problem 1: Test Cases

See the Running the program section for information.

The test cases will be in the following format:

```
radius altitude
```

For example, if the planet has a radius of 1000.5 Kilometers and desired altitude of 10.5 Kilometers, the file would have:

```
1000.5 10.5
```

Based on the diagram above, the answer would thus be `264.545`. You should only be returning the number.

Problem 2: DigBuild Log

The aliens that decided to develop *DigBuild*. Fortunately, they took a programming class, and progress is coming along nicely. One thing the game does is write to a *log* file. A log file keeps track of all events occurring in the game; such as when the game started, some information about the world, such as the position of the player, any monsters that are nearby, when the player is breaking a block or placing a block, when the player is getting attacked or attacking, etc. One of the main functions of a log is error notification. If the save file was corrupt or something the game could not handle occurs, specific details would be in the log. Since the log files contain a lot of information, it can be very difficult to read it by hand.

In order to make parsing of the log file easier, the aliens have created a helper program to parse the log file for them, and only print the important information.

The log file is designed to be in the following format. However, your program should be able to handle improperly formatted files as well.

```
[Time] [User causing action] [Event]
```

The `User causing action` will either be a username of a player or "SYSTEM". "SYSTEM" is used to represent something not caused by a user.

`Event` will be one of the following:

Event	Description
Start	The game has begun
End	The game has ended
Joined the game	The user denoted in the user field joined the game. The second line will always be an entry in this format, assuming a correctly formatted log
Left the game	The user denoted in the user field left the game
Slayed monster	The user denoted in the user field has killed a monster
Slayed [USERNAME]	The user denoted in the user field has killed a player
Was slain	The user denoted in the user field was killed by a monster or player
Has perished	The user denoted in the user field was killed by the environment or hunger
Unknown event	If the username field is SYSTEM, the game has crashed, there may be a detailed message (`Unknown event: Player disappeared`). Otherwise, the player attempted to hack the game

The first event MUST be `Start` and the last event MUST be `End` or `Unknown event` as SYSTEM.

We will be giving you the *filename* of the log file. You are to read the file and return a *String array* containing the following information:

- Index 0: Username of the person loading the file (First user to join)
- Index 1: Total number of times someone joined the game
- Index 2: "True" if the user loading the file died at least once, "False" otherwise
- Index 3: The number of monsters slain by all players
- Index 4: "No crash" if the game never crashed, the exact crash message if it did
- Index 5: "True" if the log file was formatted correctly, "False" otherwise

Given the `filename`, you must read the file and return the array as specified.

Problem 2: Function Arguments

```
parameter: filename: Name of the log file to read
```

Problem 2: Return Value

```
return: string[]: The array of strings, with the format defined above
```

Problem 2: Test Cases

Consider the following:

The test cases will be in the format defined above.

Example:

```
[07:00:00] [System] [Start]
[07:01:00] [N00b] [Joined the game]
[07:12:00] [Friend] [Joined the game]
```

```
[07:13:00] [N00b] [Was slain]
[07:15:00] [N00b] [Has perished]
[07:17:30] [Friend] [Slayed monster]
[07:20:00] [System] [End]
```

This should return the following array:

```
["N00b", "2", "True", "1", "No crash", "True"]
```

Problem 3: DigBuild Inventory

Another challenge the Aliens have faced with *DigBuild* is the inventory management system. They already have the player's inventory stored in an ArrayList/Vector, but they do not know how to write the algorithm to determine which blocks can be constructed given the player's inventory.

You will be given the filename containing the required materials to build an object. This file is a CSV (comma separated value) format, which means each element is separated by a comma. For example, a line in the file will be:

```
Wooden Plank,1,Wood
```

This means in order to build one Wooden Plank you will need 1 Wood. Note that some items will be on multiple lines. For example, to build a torch, you'd need one stick and one coal, so the lines are:

```
Torch,1,Stick
Torch,1,Coal
```

The second argument will be an ArrayList (Java) or Vector (C++) of Strings, representing the player's current inventory. You must determine what can be built according to the file, given the inventory.

Problem 3: Function Arguments

```
parameter: filename: Name of the CSV file to read (instructions to build products)
parameter: ArrayList/Vector of items in your inventory
```

Problem 3: Return Value

If the player has all the required materials to build the product (i.e. 1 stick and 1 coal to build one torch), then "Torch" should be returned.

You are to return an ArrayList of Strings (Java) or Vector of Strings (C++) of what can be built.

```
return: ArrayList<String> or Vector<String>: String list of items that can be built.
```

Problem 3: Test Cases

Each line in the test case file will be an item in the player's inventory. If the player has 3 of any item, the line will occur 3 times.

Example:

```
Coal
Stick
Stick
Stick
String
String
```

Based on the crafting instruction file, the player can now build a "Torch", or a "Fishing Rod", so you'd return:

```
["Torch", "Fishing Rod"]
```

Running the Program

Ask us for help if you are having issues. This should work in exactly the same way as the samples provided last week.

What the Aliens Need

- A solution in one of their preferred languages
  - They may not be good at coding, but they made an effort! They know a bit of C++ (using C++11) and Java (Java 11), so use one of those
- Complete code
  - They don't really know how to code, so you shouldn't leave anything for them to do.
  - In addition, you should test your code extensively to prevent bugs from showing up.
- Well-documented code
  - They want to learn the programming languages eventually, so giving them as much information about the code as possible is a must.
- Well readable code
  - The more coding conventions and standards you can follow the better. It'll make reading the code a lot easier, especially to the untrained eye.
- Fast code
  - They're accounting for having very large datasets to analyze, so the code must run quickly.
- No extra libraries
  - They don't want to be reliant on any 3rd party libraries in case they want to run it back on their planet. For this reason, only use libraries and functions that are available in your chosen language.

Using the System

You may login to the system here to find the assignments.

Once you choose what language you want to use, you can download the given starter code and work on a solution. You may use whatever IDE you may already have on your computer. If you do not have a computer with a compiler, please let us know!

A basic test case will be provided for each of the problems, however this is NOT EXTENSIVE! You MUST test your code against your own test cases. The input format for each of the 3 functions is listed under the relevant function header above.

When you are submitting, you MUST ONLY submit the file with your code. The file name is as follows:

- Java
  - AlienHelp.java
- C++
  - AlienHelp.cpp

You must submit your code to the "[Language] Submission" assignment to make sure the code compiles and the basic test cases pass. These test cases are the same as the ones given for you to test on your own computer as well.

Note, you are NOT allowed to search for the answers to these questions online, or seek help on forums, discussion boards, a friend, online tutoring services, or any other collaborative website. You MAY use the official language documentation to help with function syntax.