CSCI 136 Exam 3 Programming Part

Program 1

One of the most useful real world implementations of Stacks is with language compilers. We are going to implement a very simple version of a compiler using recursion and the Stack class. As we know, the Stack class has a push and pop. This is a built-in java.util class library. You can create an instance like this:

```
static Stack<Character> myStack = new Stack<Character>();
```

Then, you can access its methods just as like any other class.

In this program, you are going to read in any HTML file and check to see if it is valid based on its < and /> brackets. If an open curly brace "<" is found, you should push it onto the stack, and if a closed angle bracket /> is found, you should pop off the stack. At the end of the program you should print out whether the file is valid (i.e. the stack is empty) or if invalid (the stack is not empty). You are to read the file in recursively and for each line in the file you are to look recursively through the line and push and pop as necessary.

Specifics:

1. You should use the built in Stack class with its push and pop methods
2. You are to read recursively in a java file.
3. You are to read recursively in each line and push if a < open angle bracker is found and pop if a /> closed angle bracket is found.
4. You should have an appropriate exception handler for the pop.
5. After the recursive calls, you should display whether the file is valid or not.

Program 2

In this graphical program, you are going to simulate arcade chasing game. You need a single player that is movable with the arrow keys, and you need another image that will move around the toward the player. You should be able to move the player away from the object moving around the screen. You should be able to play the game for 30 seconds. (i.e. you should survive for at least 30 seconds). Don't worry about collision detection.

At the end of the simulation, you should gather the person's name, their favorite number and their email address. You should check to make sure it is correctly formatted.

Specifics:

1. Gather the person's name.
2. Gather the person's favorite number – should be a number.
3. Collect the individual's email – should have an @ and . operator.
4. Should pre-emptively check all those values as well as have exception handlers where appropriate.
5. Should have a countdown clock that starts at 30 and counts down to 0.
6. The object should move toward the player until the clock hits zero.
7. The player object should be able to move using the arrow keys.

8. Ignore collision detection.
9. For any exception handlers, you should write any exceptions to a log file.