Answer on Question #38710 - Programming - C++

If a program don't respond then more probably there is an infinite loop.

First of all you need to find out what loop is infinite. To do this you can make additional output before every loop, for example like this:

```
cout << "Entering into loop no. 1" << endl; // output before entering loop

for (...
```

Then when you run the program you will see what loop is infinite.

Once you've known what loop is infinite you can output the loop condition each time inside teh loop, for example like this:

```
for (int i = 0; i < n; i++) {

        // First line of code inside the loop

        cout << "i = " + i << ", n = " << n << endl;

        ...
```

Then you'll see how the condition of continuation is changing each iteration. Here you can see that some elements of this condition changes in other way than expected, for example in the above lines of code you see that "n" increases each time or "i" somehow decreases. That means inside the loop variables involved in the continuation condition are changed. Insert some output of this variables before and after each instruction inside the loop which can influance on them.

After you found out the place where unexpected change of those variables happens you can correct it.

Here is example:

```
for (int i = 0; i < 10; i++) {

        if (i = 3) {

                continue;

        }

        ...
```

In this example there is a loop for integer "i" from 0 to 9 except of 3. But the loop is infinite. If you check the value of "i" each time it'll be always 4 except of the first time when it will be 0. Then inserting output code before and after each line of code where "i" may change you'll see that if-condition changes it from any number (0 or 4) to 3:

```
for (int i = 0; i < 10; i++) {
        cout << "Before if-condition: i = " << i << endl;
        if (i = 3) {
                cout << "After if-condition inside brackets: i = " << i << endl;
                continue;
        }
        cout << "After if-condition after brackets: i = " << i << endl;
        ...
```

Once you've known where this happens you can analyze this instruction and find out the origin. In the example above the origin is that instead of comparison of "i" with 3 there is and assignment 3 to "i".