CSCI 135 Software Design and Analysis, C++ Lab 1

Saad Mneimneh
Hunter College of CUNY

1) write a program to output a tree like this:

```
    *
   ***
  *****
 *******
*********
```

2) write a program to output the following:

```
1+3+5+7+9=25
```

by using the expression "1+3+5+7+9" twice in different ways: once as a string, and once as an arithmetic expression. In other words, 25 should not appear anywhere in your program.

NOTE: when writing functions, there are several things that you need to take care of:

a. the name of the function
b. the number and types of parameters
c. the return type

3) write a function that takes one parameter, an integer, and returns it cubed. What should the return type of the function be? Try the function by calling it from main.

4) write a function that takes two parameters, both integers, and returns their average. What should the return type of the function be? The implementation of this function could be a bit tricky, so try it on many examples by calling it form main to make sure it works correctly. If not, try to figure out the problem.

5)
a. In the main function declare an integer x and initialize it with some value that you like. Actually, you might want to try initializing it vs. not initializing it and compare the values using cout.

b. output the location of x in memory.

c. declare a pointer p to x (think about the type) and output the value of that pointer. Compare to (b), it should be the same.

d. Now dereference p and output the result. It should be identical to x.

e. optional: what if we now want a pointer to p. What should the type of this variable be? Hint: think about the type of p and refer to the rules about pointer types. Declare that pointer, call it q.

f. optional: how can we obtain the value of x through q? Hint: Is one referencing operation enough?

6. optional: Consider the following function that takes an integers as a parameter and returns a reference to it:

```
int * pointer(int x) {
 return &x;
}


int main() {
 int x=5;
 cout<<&x<<'\n';
 cout<<pointer(x)<<'\n';
 int * p=pointer(x);

}
```

why do the cout statements output different results? In principle, what happens if we attempt to dereference p?