

C++ PROGRAM DESIGN TEST DRIVEN DEVELOPMENT

Problem Solving with Computers-I

<https://ucsb-cs16-wi17.github.io/>

C++

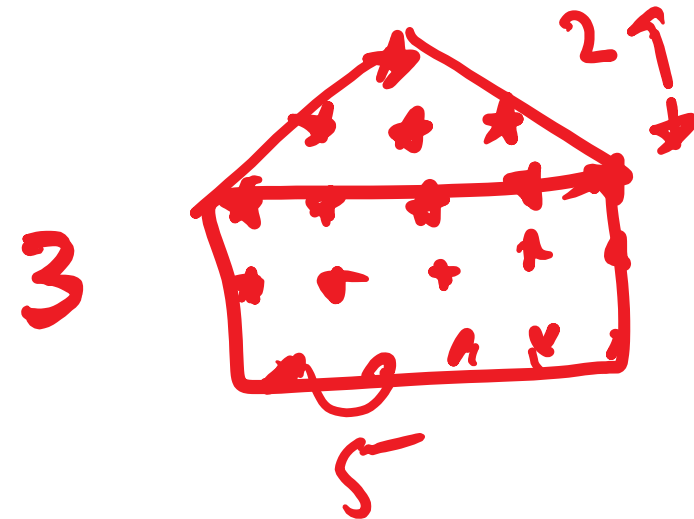
```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



ASCII Art: Write a program to draw a house of stars!

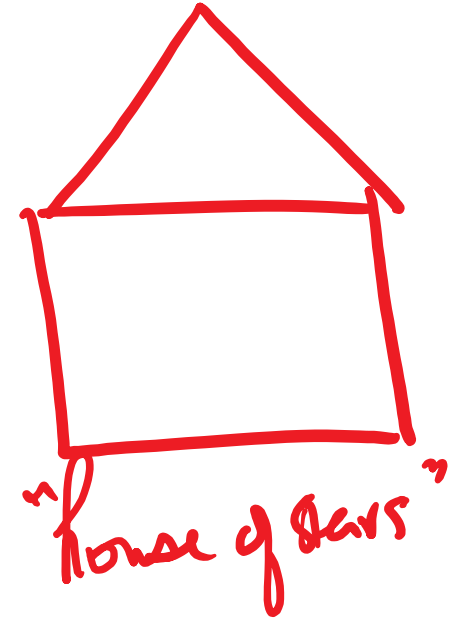
- Inputs: Dimensions of the house (width, height)
- Output: Drawing of a house made of ascii '*'



Step 1: Top-down design

- Break down the problem into subtasks

draw roof draw body



- Design your functions

return type

function_name

`string getBody(int width, int height);`

`string getRoof(int rheight);`

`string getHouse(string roof, string body);`

formal parameters



Function declarations

Step 2: Implement and test each part

- Test driven development – *Test code drives your*
- You are in control because you can.... *implementation*
 - measure your progress
 - write code systematically
 - debug systematically
 - automate testing

How to do TDD?

```
string getRoof(int rheight);
```

- Test suite: Bunch of tests
 - **Case 1:** Is `getRoof(0) == '*'`
 - **Case 2:** Is `getRoof(1) == ?` 
 - **Case 3:** Is `getRoof(2) == ?` 
- Test harness: Functions to report PASS/FAIL
If expected == actual, report **TEST PASSED!**
else report **TEST FAILED!**

testing a single function is called unit-testing

What is returned by getRoof(1)

```
string getRoof(int height){  
    int numSpaces, numStars;  
    string result=spaces=stars="";  
    for(int row = 0 ; row <=height;row++){  
        numSpaces=0;  
        numStars=1;  
        spaces=stars="";  
        for(int i=0;i<numSpaces;i++){  
            spaces+=" ";  
        }  
        for(int i=0;i<numStars;i++){  
            stars+="*";  
        }  
        result= result+spaces+stars+spaces+"\n";  
    }  
    return result;  
}
```

A

*

B

*

C

*

*

D

None of the above

Choose the replacement code to return the correct number of stars on each row for getRoof(1)

```
string getRoof(int height) {  
    int numSpaces, numStars;  
    string result=spaces=stars="";  
    for(int row = 0; row <=height; row++){  
        numSpaces=0;  
        numStars=1;  
        spaces=stars="";  
        for(int i=0; i<numSpaces; i++){  
            spaces+=" ";  
        }  
        for(int i=0; i<numStars; i++){  
            stars+="*";  
        }  
        result= result+spaces+stars+spaces+"\n";  
    }  
    return result;  
}
```

A numStars=row;

B numStars=height-row;

C numStars=2*row+1;

D numStars=2*height+1;

Choose replacement code to return the correct output for getRoof(0) and getRoof(1)

*

*

```
string getRoof(int height){
    int numSpaces, numStars;
    string result=spaces=stars="";
    for(int row = 0 ; row <=height;row++){
        numSpaces=0;
        numStars=2*row+1;
        spaces=stars="";
        for(int i=0;i<numSpaces;i++){
            spaces+=" ";
        }
        for(int i=0;i<numStars;i++){
            stars+="*";
        }
        result= result+spaces+stars+spaces+"\n";
    }
    return result;
}
```

A numSpaces=row;

B numSpaces=height-row;

C numSpaces=2*row+1;

D numSpaces=2*height+1;

Step 3: Integrate all the parts and perform black box testing

- In our example we would integrate the code for the roof and body and test the entire program with different values for width and height. For example

```
./house -1 -1
```

```
./house 3 5
```

```
.....
```

Next time

- Separate compilation with makefiles
- Pointers