FUNCTIONS
PRACTICE WITH NESTED LOOPS
TEST DRIVEN CODE DEVELOPMENT

Problem Solving with Computers-I

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

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

Clickers out – frequency AB

Infinite loops

①

```
for(int y=0;y<10;y--){
    cout<<"Print forever\n";
}
```

Initialization

update

check

Body

y is always less than 10

```
int y=0;
for(;;y++)
    cout<<"Print forever\n";
```

missing check

```
int y = 0;
for(;y<10;);
    y++;
```

; → counts as the body

y++; → outside the body of the loop

```
int y=0;
while(y<10)
    cout<<"Print forever\n";
```

→ y is not updated

Always 0

```
int y=0;
while(y=2)
    y++;
```

assignment

y==2

equality

evaluates to a number

In C++ 0 → False

everything else True

→ Used assignment instead of ==

Review: Loops

Given some integer n (may be positive or negative)
Which code that is not equivalent to the other two?

Given: n (may be positive or negative)

A.

```
for( int x = 0; x < n; x++ ) {
   cout<<x <<endl;
}
```

0
1

B.

```
int x = 0;
while(x < n) {
   cout<< x << endl;
   x++;
}
```

0
1

n=2

[✓ C.]

```
int x = 0;
do{
   cout<< x<< endl;
   x++;
} while(x < n);
```

0
1

D. They are ALL equivalent

only for n>0

Functions: Basic abstraction in programs

return type

formal params

```
int fooBar(string s, int x);
```

- Functions keep programs DRY!
- Three steps when using functions

1. DECLARE - Specify
   - name
   - inputs (and their types) (formal parameters)
   - Outputs (return type)
2. DEFINE - Actual code
3. CALL _ Run the code in the definition

Write a FUNCTION that calculates the series: 1+ 1/2+ 1/3+ ….1/n, where `n` is a parameter passed to the program

Sample run of the program:

```
$./sumseries 2
Sum of the first 2 terms is  : 1.500

$./sumseries 3
Sum of the first 3 terms is  : 1.833
```

ASCII art! Nested loops and functions

Write a FUNCTION that draws a square of a given width, and use it in a program with the following runtime behavior:

```
./drawSquare
Enter the width of the square
5
*****
*****
*****
*****
*****
```

→ use a for loop to print a single line

line = " * * * * * \n "

```
for (int i = 0; i < width; i++) {
    // print one line
    . .
}
```

Draw a triangle

Which line of the drawSquare code (show on the right) would you modify to draw a right angled triangle

```
  ./drawTriangle
Enter the length of the base
5
*
**
***
****
```

4

j	Number of stars
0	1 star
1	2 stars
2	3 stars
3	4 stars

```
void drawSquare(int side){//A

    for(int j = 0;  j < side;  j++){//B
        for(int i=0; i < side; i++){//C
            cout<<"*";
        }
        cout<<endl;
    }
    cout<<endl;

}
```

//D: A and B

//E: A and C

stars

The runtime Stack

Stack: A region in program memory to “manage” local variables

Every time a function is called, its local variables are created on the stack

When the function returns, local variables are removed from the stack

Local variables are created and deleted on the stack using a Last in First Out principle

```
int sum(int a, int b){
        cout<< a+b;
}
int main(){
        int result =0;
        int x =10, y =20;
        result = sum(x, y);
        cout<<result;
}
```

Print vs return

What is the output of the following code

```
int sum(int a, int b){
        return a+b;
}
int main(){
        int result =0;
        int x =10, y =20;
        result = sum(x, y);
        cout<<result;
}
```

Function call mechanics

What is the output of the following code

```
int sum(int a, int b){
        int result= a+b;
        exit(0);
}

int main(){
        int result =0;
        int x =10, y =20;
        result = sum(x, y);
        cout<<result;
}
```

Next time

- Automating the compilation process with Makefiles
- Intro to lab02