FUNCTIONS
PRACTICE WITH NESTED LOOPS
TEST DRIVEN CODE DEVELOPMENT

Problem Solving with Computers-I

Infinite loops

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

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

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

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

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

Review: Loops

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

A.

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

B.

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

C.

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

D. They are ALL equivalent

Functions: Basic abstraction in programs

- Functions keep programs DRY!
- Three steps when using functions
  1. DECLARE
  2. DEFINE
  3. CALL

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

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

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

```
 5 void drawSquare(int side){//A
 6
 7     for(int j = 0;  j < side;  j++){//B
 8         for(int i=0; i < side; i++){//C
 9             cout<<"*";
10         }
11         cout<<endl;
12     }
13     cout<<endl;
14
15 }
```

//D: A and B

//E: A and C

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;
}
```