Discussion Section
Week 7

Recursion

What is recursion?

- Calling a function within itself
- **Base Case**
- Idea is to break a problem down into its simplest forms, and build the solution back up

Example

```cpp
unsigned long int rec_suffix_sum(unsigned int n, unsigned int s);

int main(void)
{
    std::cout << rec_suffix_sum(10, 5) << std::endl;
    return 0;
}

unsigned long int rec_suffix_sum(unsigned int n, unsigned int s){
    //Base case when only the last number is wanted, returns that number
    if (s == 1){
        return n;
    }

    //Add together suffix sums until s reaches the base case
    return n + rec_suffix_sum(n - 1, s - 1);
}
```

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

derek@DESKTOP-3L8T6AU:/mnt/c/Users/Derek Jacobs/Desktop/CSC/TA/211$ g++ test.cpp && ./a.out
40
```

Where is it useful?

- Some algorithms are meant to be recursive
  - Fibonacci
- Sometimes it’s just easier
  - Binary Search

Check if a value is in a sorted array (Binary Search)

- Given an array A, the size of that array n, and a search key k, check if
  - Base Case:
    - Array is size 1
      - Check if the only element in A is equal to k
      - Return true or false based on that
  - If the element in the middle of the array is lesser than the key, search the right half
  - Otherwise if the middle element is greater, search the left half
  - Else if they're equal, you've found your key, return true

Exercise 1 (10 min)

Implement Binary Search, test with the following input: A = [0,1,2,3,4,5,6,7,8,9,10], n = 11, k = 2

Given an array A, the size of that array n, and a search key k, check if

- Base Case:
  - Array is size 1
    - Check if the only element in A is equal to k
    - Return true or false based on that
- If the element in the middle of the array is lesser than the key, search the right half
- Otherwise if the middle element is greater, search the left half
- Else if they're equal, you've found your key, return true

Solution

```
bool rec_bin_search(const int *A, unsigned int n, int k){
    if (n == 0){
        return false;
    } else if ((n == 1) && (k == *A)){
        return true;
    } else if ((n == 1) && (k != *A)){
        return false;
    }

    if (A[n / 2] > k){
        rec_bin_search(A, (n/2), k);
    } else if (A[n/2] < k){
        A += (n/2);
        if ((n % 2) == 0){
            rec_bin_search(A, (n/2), k);
        } else {
            rec_bin_search(A, (n/2) + 1, k);
        }
    } else if (A[n/2] == k){
        return true;
    }
}
```

Iterative vs Recursive

- Anything that can be solved recursively can be solved iteratively and vice versa

Exercise 2 (5 min)

Iteratively, do the following:

Multiply all elements in an array together

Solution

```
int multElements(int *arr, int size)
{
    int product = 1;
    for(int i = 0; i < size; ++i) product *= arr[i];

    return product;
}
```

Exercise 3 (10 min)

Solve the same problem, but recursively

Solution

```
int multElements(int *arr, int size)
{
    if (size == 1) return arr[0];
    else return *arr * multElements(arr+1, size-1);
}
```

Backtracking

- Trying a solution until it doesn’t work
- Back up until it works

Example

Solving Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Problems with recursion

- Complexity
  - Time and Space

```cpp
int fib(int n)
{
    std::cout << "Recursive Call";
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n -1) + fib(n-2);
}
int main(void)
{
    fib(10);
    std::cout << std::endl;
    return 0;
}
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: wsl

```
derek@DESKTOP-3L8T6AU:/mnt/c/Users/Derek Jacobs/Desktop/CSC/TA/211$ g++ test.cpp && ./a.out
Recursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecur
sive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive
CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallR
ecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecurs
ive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive C
allRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRe
cursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursi
ve CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Ca
llRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRec
ursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursiv
e CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Cal
lRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecu
rsive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive
 CallRecursive CallRecursive CallRecursive Call
```

```
}
int main(void)
{
    fib(100);
    std::cout << std::endl;
    return 0;
}
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: wsl

```
CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallR
ecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecurs
ive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive C
allRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRe
cursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursi
ve CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Ca
llRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRec
ursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursiv
e CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Cal
lRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecu
rsive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive
 CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Call
Recursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecur
sive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive
CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallR
ecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecurs
ive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive C
allRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRe
cursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursi
ve CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Ca
llRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRec
ursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursiv
e CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Cal
lRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecu
rsive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive
 CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Call
Recursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecur
sive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive
CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallR
ecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecurs
ive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive C
allRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRe
cursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursi
ve CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive Ca
llRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallRecursive CallR
```

Final Remarks

- Recursion is neat (but tricky)
- Anything you can do iteratively, you can also do it recursively
- Be weary of the complexity of your programs