Week 4

Announcements

- Congrats on getting through midterm 1!
- Project 3
    - 11/3 (next tuesday!) 11:00 PM
    - zybook exercises

N and a Half

```
// Not the cleanest loop
getline
loop
  some break condition
  getline

// Better way to write the above
while(true) {
  getline
  check break condition
}

// *** Exercise ***
// output 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
// with two separate cout's

int i = 1;

while (true) {
  cout << i;
  if (i >= 10)
    break;
  cout << ", ";
  i++;
}

// Other Syntax?
for(;;) {
  cout << i;
  if (i >= 10)
    break;
  cout << ", ";
  i++;
}
```

Char vs String

```
// Which ones are valid?

char c1 = 'x';
char c2 = "x"; // not valid
char c3 = '';  // not valid
char c4 = 33;      // in ASCII, '!'
char c5 = c4++;

char a = 'a';
char c = (a + 2) // c will be assigned to 'c'

bool b1 = ('A' < 'a') // true in ASCII

// ASCII 48 == '0'
// ASCII 50 == '2'

string s1 = "Hey there!";
string s2 = 'x'; // not valid
string s3 = "";
s3 += 'a';           // s3 = "a"
s3 += s1;            // s3 = "aHey there!"
string s5 = 44   // not valid

// 2 ways to append strings
string greeting = "Salutations ";
string name = "Taasin";

greeting.append(name);
greeting += name;

// Difference between .at(0) and [0]
// Square brackets are faster, but will give you random values if you're
// out of bounds
// the function does some error checking for you, but is slower
string s = "abc";
s.at(0) = 'a';
s[0] = 'a';
```

Functions

```
// function prototype
// return types include: void, int, string, bool
void f1(int i, string s);

// function implementation
int f2(int i, string s) {
  return 5;
}

int r = f1(5, "s"); // error, return type's don't match


// clean up the style here
bool func() {
  // ...
  if (numDigits == 10)
    return true;
  else
    return false;

  // preferred syntax
  return (numDigits == 10);
}

// function stubs, remember to develop incrementally!

// need function prototype if you define the function below where it is used
int f2();

int main() {
  int g = f2();
}

int f2() {
  return 5;
}
```

Handling Strings

```
// Which Library?
#include <cctype>

// Predicates
isdigit, isupper, islower, isalpha, etc.
int i = isdigit('9')

// Camel Case: isPhoneNumber(), Underscores: is_phone_number()
```

```
tolower, toupper, etc.

char a = tolower('A');

string s = "ABC";
for (int i = 0; i != s.size(); i++ ) {
  s.at(i) = tolower(s.at(i));
}

// to change the character, you need to assign it
s.at(0) = tolower(s.at(0));

// what if its the empty string? will throw an error
string s = "";
tolower(s.at(0))
```

Pass by Value vs Reference

```
// Pass by Value
int square1(int a) {
  return a*a;
};

// Pass by Reference
void square2(int& a) {
  a = (a*a);
}

int a = 5;

int value = square1(a);
cout << value << endl; // outputs
cout << a << endl;     // 5

square2(a);
cout << a << endl; // outputs 25
```

Clarifications

```
char c3 = '';  // can't do this
char c4 = '\0'; // null character
```

```
// uppercase letters have lower ASCII values than lowercase ones
bool b1 = ('A' < 'a') // true in ASCII

int i = isdigit('9')
// bool b = isdigit('9')

// for pass by reference, you can put the ampersand anywhere between the type
// and the name
// all of these are valid
int square1(int& a);
int square1(int &a);
int square1(int & a);
```

Incremental Development

```
// From worksheet 4, just an example of how I would develop this function
// incrementally

// ********************************************
// I'd start by writing the function prototype
// Inputs and outputs are important to get right
bool isPalindrome2(string s);

// ********************************************
// Then handle the most trivial cases, which are the empty string
// or strings of one character
// it always returns true, which is fine for now (later I will return false
// in the middle if I figure out that it isn't a palindrome)
bool isPalindrome2(string s) {
  // base cases
  if (s.size() == 0 || s.size() == 1)
    return true;

  return true;
}

// ********************************************
// next I'd write a function to remove the spaces in the string
string noSpaces(string s) {
  // return s but without spaces
  string temp;
  for (int i = 0; i != s.size(); i++) {
      if ( !isspace(s.at(i)) )
          temp += s.at(i);
  }
  return temp;
}
```

```
bool isPalindrome2(string s) {
  // base cases
  if (s.size() == 0 || s.size() == 1)
    return true;

  string sNoSpaces = noSpaces(s);
  // use cout to see your incremental results
  cout << s << " * " << sNoSpaces << endl;

  return true;
}

// ********************************************
// and finally, add the loop to iterate through the string and compare characters
string noSpaces(string s) {
  // return s but without spaces
  string temp;
  for (int i = 0; i != s.size(); i++) {
      if ( !isspace(s.at(i)) )
          temp += s.at(i);
  }
  return temp;
}

bool isPalindrome2(string s) {
  // base cases
  if (s.size() == 0 || s.size() == 1)
    return true;

  // remove spaces
  string sNoSpaces = noSpaces(s);

  // compare character in 1st half of string with corresponding one in 2nd half
  // think about how this loop works for even and odd length strings
  for (int i = 0; i != sNoSpaces.size()/2; i++) {

      // make sure you understand why its comparing .at(i) to .at(size() -1 -i)
      if ( sNoSpaces.at(i) != sNoSpaces.at(sNoSpaces.size()-1-i) )
        return false;
  }

  return false;
}
```