1 Strings in C

C++ has a `string` type defined in the <string> header file, and that you can use a C++ `string` almost like a basic type. The `string` type lives in the `std` namespace.

```
std::string myString = "Hello, World";
cout << "The value of myString is " << myString << endl;
```

Unlike C, C++ provides a string type as a first-class data type. C++ contains several functions from the C language that operate on strings, defined in the <cstring> header. These functions do not handle memory allocation. For example, the `strcpy()` functions takes two strings as parameters. It copies the second string onto the first, whether it fits or not. The following code attempts to build a wrapper around `strcpy()` that allocates the correct amount of memory and returns the result, instead of taking in an already allocated string. It uses the `strcpy()` function to obtain the length of the string. The caller is responsible for freeing the memory allocated by `copyString()`.

```
char* copyString(const char* str)
{
  char* result = new char[strlen(str) + 1];
  strcpy(result, str);
  return result;
}
```

1.1 String Literals

You've probably seen something like this

```
cout << "hello" << endl;
```

In the preceding line, `"hello"` is a *string literal* because it is written as a value, not as variable. Memory associated with a string literal is in a read-only part of memory. Allowing the compiler to optimize memory usage by reusing references to equivalent string literals. This is called *literal pooling.*

1.2 The C++ string class

C++ provides a much-improved implementation of the concept of a string as part of the Standard Library. In C++ `std::string` is a class that supports many of the same functionalities as the <cstring> functions, but takes care of memory allocation for you.

```
string A("12");
string B("34");
string C;
C = A + B; // C will become "1234"
```

Another problem with C strings is that you cannot use == to compare them. With C++, `==, !=, <, >` etc, are all overloaded to work on the actual string characters.

1.3 Numeric Conversions

The `std` namespace includes a number of helper functions to convert numerical values into strings and vice versa.

```
string to_string(int val);
string to_string(unsigned val);
string to_string(long val);
string to_string(unsigned long val);
string to_string(long long val);
string to_string(unsigned long long val);
string to_string(float val);
string to_string(double val);
string to_string(long double val);
string to_string(int stoi(const string& str, size_t *idx=0, int base=10));
string to_string(long double val);
string to_string(long stol(const string& str, size_t *idx=0, int base=10));
string to_string(unsigned long stoul(const string& str,
                                     size_t *idx=0, int base=10));
string to_string(long long stoll(const string& str, size_t *idx=0, int base=10));
string to_string(unsigned long long stoull(const string& str,
                                           size_t *idx=0, int base=10));
string to_string(float stof(const string& str, size_t *idx=0));
string to_string(double stod(const string& str, size_t *idx=0));
string to_string(long double stold(const string& str, size_t *idx=0));
```