Basics of C++

1.1 Summary
C++ is an extension of C. So the simplest program just has a main function. The program is compiled on our system with g++, which by default produces an executable a.out that is run from the current directory.

C++ has for, while and do for loops, if and switch for conditionals. The standard output is accessed by cout. The standard input is accessed by cin. These require inclusion of iostream library. The language is case-sensitive.

1.2 Data Types
C++ has several data types that can be used to store integers; we will mainly use int. We will use char for characters. Note that, one can treat a char as an integer for arithmetic: for example

```cpp
char myChar = 'D';
int pos = myChar - 'A' + 1;
cout << "the char " << myChar << " is in position " << pos
    << " in the alphabet" << endl;
```

These integer-types also come in unsigned versions. We will not use these much. But do note that arithmetic with unsigned data types is different. For example the code

```cpp
for( unsigned int X=10; X>=0; X--) cout << X;
```

is an infinite loop, since decrementing 0 produces a large number.

C++ has several data types that can be used to store floating-point numbers; we will almost always use double. There is also bool for boolean (that is, true or false); sometimes integers are substituted, where 0 means false and anything non-zero means true.

1.3 Arrays
Arrays in C++ are declared to hold a specific number of the same type of object. The valid indices are 0 up to 1 less than the size of the array. The execution does no checking for references going outside the bounds of the array. Arrays can be initialized at declaration.
1.4 Functions

A function is a self-standing piece of code. It can return a variable of a specified type, or have type void. It can have arguments of specific type. In general, variables are passed by value, which means that the function receives a copy of the variable. This is inefficient for large objects, so these are usually passed by address (such as automatically occurs for arrays) or by reference (discussed later).

To aid the compiler, a prototype of a function at the start of a program tells the compiler of the existence of such a function: it specifies the name, arguments, and type of the function. The actual names of the arguments are optional, but recommended.

1.5 Pointers

A pointer stores an address. A pointer has a type: this indicates the type of object stored at the address to which the pointer points. A pointer is defined using the *, and is dereferenced thereby too. An array name is equivalent to a pointer to the start of that array. Primitive arithmetic can be applied to pointers. To indicate that a pointer points to nothing, it is set equal to nullptr.

1.6 Strings

There are two options to store strings in C++. The first is the way done in C, now called C-strings. A C-string is stored as a sequence of chars, terminated by the null character (which is denoted '\0' and has value 0 as an int). The user must ensure that the null terminator remains present. Constant strings defined by the user using quotation marks are automatically C-strings. With the cstring library, strings can be compared, cin-ed and cout-ed, copied, appended, and several other things. C-strings are passed to functions by reference: that is, by supplying the address of the first character using the array name or a char pointer.

We will mostly use the object from the string class provided in the string library. These can be compared, cin-ed and cout-ed, assigned C-string, appended, etc.

Sample Code

The first example code prints out the prime numbers less than 100. We will explain the use of namespace's later.

In the second example code, the binarySearch function searches a sorted array for a specific value. It returns the index if it finds the value, and -1 otherwise.

```
primality.cpp
BinarySearch.cpp
```