CSCI 159 Exam Quick Reference Sheet

C++ operators   // single line comments come after the two slashes

Arithmetic operators:	`+   -   *   /   %   ++  --`
Assignment operators:	`=  +=  -=  *=  /=  %=`
Boolean operators:	`&&  ||  !  ==  !=  <=  >=  <  >`

Data Types

Data	Keywords	Literal Examples	Special values
integers:	short, int, long	3, -200, 0	INT_MAX, INT_MIN (climits library)
reals:	float, double	3.14, -0.0003	FLT_MAX, FLT_MIN (cfloat library)
character:	char	'x'	`\'  \"  \\  \t \n  \0`
boolean:	bool	true, false	

Sample variable declarations (with/without initialization)

```
int    i;       int    i = 3;
char   c;       char c = 'Q';   char c = '\n';
bool   b;       bool b = true;
long arr[5];    long arr[5] = { 0, 0, 0, 0, 0 };   // array assignment only valid
char str[10];   char str[] = "some text";          //    at point of declaration
```

Sample constant declarations

```
const double Pi = 3.1415;
const char*  ErrMsg = "Error: something terrible happened!\n";
const char[] ErrMsg = "Error: something terrible happened!\n"; // works like char*
```

Sample enumerated type definitions

```
enum Weekdays { Sun, Mon, Tue, Wed, Thu, Fri, Sat };
enum Commands { Quit = 'Q', Continue = 'C', Print = 'P' };
```

Sample input with cin (iostream library, namespace std)

```
cin >> x;   cin >> (noskipws) >> x;

// example of checking for input failure,
//    and, on failure, read/discard up to N characters or until end-of-line
if (cin.fail()) {
   cin.ignore(N, '\n');
   cin.clear();
}
```

Sample output with cout (iostream library, namespace std)

```
cout << "x is " << x << endl; // display text, variable, and newline
```

setting fixed width and precision (iomanip library, namespace std)

```
setiosflags(ios::fixed);
cout << setprecision(2) << x; // show exactly 2 digits after decimal place
cout << setw(5) << x; // pad x (with spaces) to width at least 5
```

The C++ string class (using the <string> library, namespace std)

```
string str; // declare a string variable str
str = "blah blah blah"; // assign text to a string
str[3] = 'x'; // change the fourth character in the string to x
str.c_str() // get as a char[], null-terminated string
```

Other useful library functions and constants

cctype	cfloat	cmath
bool isalpha(char)	FLT_MIN, FLT_MAX	double ceil(double)
bool isalnum(char)	DBL_MIN, DBL_MAX	double floor(double)
bool isdigit(char)		double fabs(double)
bool islower(char)	climits	double log(double)
bool isupper(char)	-------	double pow(double, double)
bool ispunct(char)	CHAR_MIN, CHAR_MAX	double cos(double)
bool isspace(char)	SHORT_MIN, SHORT_MAX	// also acos, sin, asin, tan, atan

```
char tolower(char)     INT_MIN, INT_MAX         double sqrt(double)
char toupper(char)     LONG_MIN, LONG_MAX

cstring                              cstdlib
-------                              -------
char[] strcat(char[],  char[])       int abs(int)
char[] strncat(char[], char[], int)  int atoi(char[])
char[] strcpy(char[],  char[])       float atof(char[])
char[] strncpy(char[], char[], int)  void srand(time(NULL)) // needs ctime lib
int    strcmp(char[],  char[])       int rand(int)
int    strncmp(char[,] char[], int)
int    strlen(char[]

Sample control structures
=========================
if (expr) {                  // works on short, int, long,      for (x = 1; x < 9; x++) {
   .......                   //    char, or enum values             ....
} else if (expr) {           switch (expr) {                    }
   ........                     case value1:
} else {                            .....                       while (x < 9) {
   ........                         break;                          ....
}                               case value2:                        x++;
                                    .....                       }
// is X between 3 and 9?            break;
if ((3 < X) && (X < 9)) {       default:                        do {
   // yes it is                     .....                           ....
} else {                            break;                          x++;
   // no it isn't            };                                 } while (x < 9);
}
         Sample function prototypes and implementations         Sample calls
         ==============================================         ============
void swap(int &a, int &b);      float calc(int x, float f)      int main()
......                          .....                           {
void swap(int &a, int &b)       float calc(int x, float f)         int i = 1;
{                               {                                  int j = 2;
    int temp = a;                  float result = x * f;           swap(i, j);
    a = b;                         return result;                  float f = calc(i, 2.5);
    b = temp;                   }                                  int array[20];
}                                                                  initArray(array, 20);
                                                                }
Pointer examples
================
int i;       // an integer variable i
int *iPtr;   // iPtr can point at integers in memory
iPtr = &i;   // iPtr now points at variable i (& takes the address of i)
(*iPtr) = 3; // store 3 whereever iPtr points in memory

Function prototype with a pointer passed by ref        Basic bubblesort algorithm
===============================================        ==========================
void doSomething(int* &ptr);                           for passNum = 1 to N-1
                                                           for pos = 1 to N-1
Dynamic memory allocation and deallocation                     if arr[pos] < arr[pos-1]
==========================================                        swap(arr[pos-1],arr[pos])
using new/delete
----------------
int *i = new (nothrow) int;         // alloc single int
delete i;                           // free the int
float *f = new (nothrow) float[10]; // alloc arr of floats
delete [] f;                        // free the array
// plus test for nullptr after any call to new

Sample struct definition and use                        Basic binary search algorithm
===================================                     =============================
struct Info {             Info i;                       low = 0
   char initials[2];      i.id = 0;                     high = arraySize-1
   int id;                i.value = -34.216;            while low <= high
   float value;           i.initials[0] = 'D';             mid = (low+high)/2
};                                                         if target==arr[mid]
                                                              return mid (found it!)
// using pointers to structs                               else if target < arr[mid]
Info* ptr = new (nothrow) Info;                               high = mid-1
if (ptr != nullptr) {                                      else
   ptr->id = 10;   // using -> notation                       low = mid+1
   (*ptr).id = 10; // using *. notation                 return -1 (never found it)
}
```