ECOR 1606 Final Lab Test Crib Sheet and Hints (Note: 2 pages long)

CONTROL STRUCTURES

simple if:

```
if (boolean exp) {
        statements // body
}
```

if-then-else:

```
if (boolean exp) {
        statements // true part
} else {
        statements // false part
}
```

multi-way if:

```
if (boolean exp) {
        statements // part 1
} else if (boolean exp) {
        statements // part 2
} else if (boolean exp) {
        statements // part 3
… // and so on
} else { // else part optional
        statements // else part
}
```

while loop (pre-test):

```
while (boolean exp) {
        statements // body
}
```

do-while loop (post-test):

```
do {
        statements // body
} while (boolean exp);
```

for loop:

```
for (exp1; exp2; exp3) {
        statements // body
}
```

is equivalent to:

```
exp1;
while (exp2) {
        same statements
        exp3;
}
```

break statement:

```
break;
```

– causes an exit from the enclosing loop.

continue statement:

```
continue;
```

– sends control back to the top of the enclosing loop.

CALL BY …

```
int sample (int a, int &b);
```

"a" is call-by-value (just like a regular variable but given a value when the function is called).

"b" is call-by-reference. all operations on "b" actually operate on the variable supplied.

MODEL PROGRAM

```
#include <iostream>
#include <cmath>
#include <iomanip>
#include <cstdlib>

using namespace std;

int add(int x, int y) {
    int result;
    result = x + y;
    return result;
}

int main() {
    int a, b, c;
    cout << "Enter two values: ";
    cin >> a >> b;
    c = add(a, b);
    cout << "The answer is " << c << endl;
    system("PAUSE");
    return 0;
}
```

EXPRESSIONS

`<` is less than

`>` is greater than

`<=` is less than or equal to

`>=` is greater than or equal to

`==` is equal to

`!=` is not equal to

`||` OR (either side is true)

`&&` AND (both sides are true)

`!` NOT (changes true to false, false to true)

`%` modulus (gives remainder from division)

`X++` means "use value of X, then increment X"

`X--` means "use value of X, then decrement X"

`++X` means "increment X, then use new value"

`--X` means "decrement X, then use new value"

`X += Y` is equivalent to `X = X + (Y)`

(same idea for `-=`, `*=`. and `/=`)

OUTPUT (use iostream, iomanip)

```
cout << showpoint;   // forces the display of decimal point and zeroes (in no fractional part)
                     // to the right of the decimal point. this remains in effect until changed.
cout << fixed;       // forces use of non-scientific notation. this remains in effect until changed.
cout << setprecision(value); // selects the number of digits to be displayed to the right of the decimal point
                     // when outputting double values. the choice remains in effect until changed.
cout << setw (value) << … // indicates that the next value output is to occupy the specified number of
                     // columns. a one shot deal – affects only the next value output
cout << setfill(ch); // selects the fill character to be used when padding output values to a specified width.
                     // the choice remains in effect until changed.
```

LIBRARY FUNCTIONS

```
double fabs(double x);   returns the absolute value of "x", for real numbers
int abs(int x);          returns the absolute value of "x", for integers
double log (double x);   natural log (log base e)
double log10(double x);  log base 10
double exp(double x);    returns "e" to power of "x"
double sqrt(double x);   returns the square root of "x"
double pow (double x, double y);   returns "x" to the power of "y"
double sin(double x);    returns the sine of "x" (note: "x" is in radians)
double cos(double x);    returns the cosine of "x" (note: "x" is in radians)
double asin(double x);   returns the inverse sin of "x" (in radians)
double acos(double x);   returns the inverse cosine of "x" (in radians)
double sinh(double x);   hyperbolic sin
double cosh(double x);   hyperbolic cosine
```

Reminder: When **using** a function, you do **not** include the types (e.g. double, etc.).

Seven Deadly Lab Test Sins

Stuck? Check that you aren't making any of the following mistakes:

1/. Integer division.

```
(1 / 5) is zero!
```

2/. Assignments in conditions:

```
if (a = b) { // a disaster
```

3/. Legal (but nonsensical) logical expressions:

```
if (7 < a < 67) {
if ((a || b || c) == 0) {
```

4/. Variables with the same name as a function.

```
sin = sin(x);
error: 'sin' cannot be used as a function
```

5/. Missing multiplication operators in expressions.

```
a = b(4 + 7);
error: 'b' cannot be used as a function
```

6/. Variables that are not properly initialized.

7/. Improperly chosen functions.

do you have *abs* where you want *fabs*?

do you have *log* and where you need *log10* (or vice versa)?