PASS MOCK EXAM – FOR PRACTICE ONLY

Course: **ECOR 1606** Facilitator: **Micah Klettke**
Dates and locations of mock exam take-up:

Thursday, Apr. 11	**12-4pm**	**ME 3275**
Friday, Apr. 12	**2-6pm**	**ME 4499**

IMPORTANT:

It is **most beneficial** to you to write this mock final **UNDER EXAM CONDITIONS**. This means:

- Complete the mock final in 3 hour(s).
- Work on your own.
- Keep your notes and textbook closed.
- Attempt every question.

After the time limit, go back over your work with a different colour or on a separate piece of paper and try to do the questions you are unsure of. Record your ideas in the margins to remind yourself of what you were thinking when you take it up at PASS.

The purpose of this mock exam is to give you practice answering questions in a timed setting and to help you to gauge which aspects of the course content you know well and which are in need of further development and review. Use this mock exam as a *learning tool* in preparing for the actual exam.

Please note:

- Come to the PASS session with your mock exam complete. There, you can work with other students to review your work.
- Often, there is not enough time to review the entire exam in the PASS session. Decide which questions you most want to review – the facilitator may ask students to vote on which questions they want to discuss.
- Facilitators do not bring copies of the mock exam to the session. Please print out and complete the exam before you attend.
- **Facilitators do not produce or distribute an answer key for mock exams**. Facilitators help students to work together to compare and assess the answers they have. If you are not able to attend the PASS session, you can work alone or with others in the class.

Question 1

Given the following function:

```
int q1 (int x[], int &y, int z){
for (int i = z; i >= 0; i--){
    x[i] = y++;
}
z += 1;
return z - 2;
}
```

a) What will be the output if the following code is executed?

```
int b = 0, c = 4, d;
int a[5] = {0};
d = q1(a, b, c);
for (int i = 0; i <= c; i++){
    cout <<  a[i] << " ";
}
cout << endl;
```

b) What will be the output if the following code is executed?

```
int b = 2, c = 3, d;
int a[5] = {0};
d = q1(a, b, c);
c *= 0.8;
cout << b % 4  << " " << c << " " << d*1.25 << endl;
```

c) What will be the output if the following code is executed?

```
int b = 9, c = 2, d;
int a[5] = {0};
if ((b % c == 1) || ((a[1] + 3 * 2 % 2 == 0)
     && (a[c] / c == b + 2 * a[b]))){
      b = 2;
      c = 1;
}
else {
     b = 1;
     c = 2;
}
d = q1(a, b, c);
cout << b << " " << c << " " << d << endl;
```

d) What will be the output if the following code is executed?

```
double a = 1./3;
    for (int i = 0; i <= 6; i+=2){
        cout << setfill('X') << setprecision(i) << setw(i+2) <<
++a << endl;
    }
```

Question 2

a) Write a function that takes an array of integers and outputs two **sorted** arrays: one that contains all the even integers, and one that contains all the odd integers. You may use the following function that sorts an array.

```
void sortArray(int array[], int size);
```

b) Write a function that, given an array of positive integer values and size of the array, returns how many multiples of the minimum value exists in the array.

Examples:
array = {5, 6, 3, 7, 9, 12} function should return 3 (6, 9, 12 are multiples of 3)
array = {3, 7, 2, 5} function should return 0 (no multiples of 2}

Question 3

a) Write a function that calculates $\sin(x)$ using the Taylor series expansion which is defined as:

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$$

Your function should also input the number of terms to calculate to and it should output the relative error of the calculated value (You may use the built-in function `sin(x)` ONLY for calculating the relative error).

b) Now write another function that uses the function that you just wrote to calculate $sinc(x)$ to a given maximum relative error (Hint: as the number of terms in the Taylor series increases, the relative error decreases).

$$sinc(x) = \frac{\sin(x)}{x}$$

Are there any special values you should check for?

Question 4

a) Given the following snippet of code (and assuming the filestream fin was initialized previously in the code).

```
int input, sum = 0;
for(;;){
   fin >> input;
   if (fin.fail()){
      cout << "FILE ERROR: Unexpected Input\n";
      system("PAUSE");
      return 0;
   }
   if (fin.eof()){
      break;
   }
   if (input % 3 != 0){
      sum += input;
   }
}//end for
cout << "The sum is: " << sum << endl;
```

What will the output be if fin points to a file that contains the following:

```
1 12 7 6 2
3     3    4
```

Is there anything wrong with the above code? If so, state why and propose a solution.

b) You are given a file of integers (data.txt) that contains corrupted data and are asked to remove the corrupted data while preserving as much of the original contents as possible. Some analysis has been performed on the file that reveals the maximum width of an error is 3 characters.

You are to write a program that takes the file “data.txt” and outputs a file “data_good.txt” that contains the contents of the original file. This file should contain 4 integers per line, with fixed spacing of 4 characters per integer. Finally, the program should output the percentage of data recovered (in terms of integers recovered, not per character)

An example of input and output data is shown below:

data.txt

```
14 333 X 24 2 182 XXX
444 XX 843 47 X XXX 545
3
X 22 XXX 9
```

data_good.txt

```
  14 333  24   2
 182 444 843  47
 545   3  22   9
```

console:

```
Percentage of data recovered: 63.2%
```

Question 5

State what is wrong with the following snippets of code (assume all variables and file streams are setup and initialized properly where applicable). Fix the errors, re-writing the code if necessary.

a) calculates the sum of numbers stored in a file

```
sum = 0;
while (!fin.eof() | !fin.fail()){
    fin >> input;
    sum = sum + input;
}
cout << sum << endl;
```

b) opens file located at `files\data01\data040613.txt` and checks for errors

```
ifstream fin;
fin.open("files\data01\data040613.txt");
if (fin.fail()){
      cout << "FILE ERROR: Unexpected Input\n";
      fin.ignore(MAX_INT,'\n');
      fin.clear();
}
```

c) asks user for filename and checks for errors

```
cout << “Enter a filename: “;
cin >> filename;
while (cin.fail()){
      cout << "Error opening file\n";
      cin.clear();
      cout << “Enter a filename: “;
      cin >> filename;
}
fin.open(filename);
```

ECOR 1606 Final Exam Crib Sheet

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 { // an else part is
            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 int c[]);
```

"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.
"c" is call-by-reference. all operations on "c" actually operate on the array supplied.

ARRAYS

```
// sample declaration
int a[4] = {1, 2, 4, 12 };

// typical use
sum = 0;
for (i = 0; i < array_size; i++) {
   sum += a [i];
}

// typical function
void write_array(int a[], int n) {
   int i;
   for (i = 0; i < n; i++) {
      cout << a[i] << endl;
   }
}
```

MODEL PROGRAM

```
#include <iostream>
#include <cmath>
#include <iomanip>
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 /=)

INPUT (use iostream, fstream)

```
ifstream xin;       // declares an input stream object (for reading files)
xin.open(string);// attaches the input stream object to the file specified
xin.fail()          // returns true if the stream is in the failed state (something's gone wrong)
xin.eof()           // returns true if the program has tried to read past the end of the file
xin.clear()         // resets the failure flag
xin.ignore(count, ch);     // discards input characters until "count" characters have been discarded
                           // or character "ch" has been discarded (whichever comes first)
```

OUTPUT (use iostream, fstream, iomanip)

```
ofstream xout;  // declares an output stream object (for reading files)
xout << setiosflags(ios::fixed|ios::showpoint);  // forces use of non-scientific notation and the display of zeroes
                                                 // to the right of the decimal point. this remains in effect until changed.
xout << 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.
xout << 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
xout << setfill(ch);               // selects the fill character to be used when padding output values to a specified width.
                                   // the choice remains in effect until changed.
xout.fill();                       // returns the current fill character.
```

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
```