# ECOR 1606 D Winter 2014 Midterm Exam

## Sample Solutions

| Q1 (/10) | Q2 (/7) | Q3 (/6) | Q4 (/10) | Q5(/12) | Total (/45) |
|---|---|---|---|---|---|
| | | | | | |

**Instructions:**

- The test is 75min long (1 hour 15 minutes).
- There are five questions worth 45 marks on 11 pages.
- The last page of the test gives the syntax for the C--/C++ statements studied to date.
- You may remove the last page from the test and use the back of it for rough work.
- All code required for this test is to be written in C--/C++.
    - Pseudo code is not acceptable, e.g. you cannot use words like "and" and "or".
    - You must include brackets and semi-colons as per valid C++ syntax.
    - You may **only** use the C--/C++ statements listed on p.11. Note that the only variable type permitted is **double** (no **int** variables allowed).
    - You do **not** need to write any of the C++ code from "framework.cpp", e.g. #include's, int main(), return 0, etc.
- Your answers to the programming questions (#1 and #5) must include:
    - variable declarations
    - comments
    - proper indentation
- Do not ask a question unless you believe that you have found a typo.

**Question 1 (10 marks):**

Write a program that reads in two numbers and, if the input is valid, outputs 2 times sum of every second integer that lies between the two values (**excluding** the values themselves). If either number is not an integer or is not positive, or if the first number is not less than the second number, just output an error message. The sample runs below should give the idea. User inputs are in **bold**.

**Important Notes:** Your program should use a **loop** to calculate the sum, i.e. you will lose marks if you use a formula to calculate the sum. Your program must **not** have a **sentinel** loop. It should just deal with just **one** set of inputs and stop.

**Hint:** For full marks, you must use function *isInt*. Remember that *isInt(x)* returns true if *x* is an integer and false otherwise.

**First example (the output is twice the sum of 6 + 8 + 10):**
Enter two positive integers (smaller number first): **5 11**
Double the sum of every second integer in between these values is: 48

**Second example (the first number is not less than the second number):**
Enter two positive integers (smaller number first): **10 6**
You did not follow the instructions!

**Third example (one or both values are not integers):**
Enter two positive integers (smaller number first): **6.5 10**
You did not follow the instructions!

**Fourth example (one or both values are not positive):**
Enter two positive integers (smaller number first): **-1 0**
You did not follow the instructions!

**Fifth example (there are no numbers between the two values, so the output is 0):**
Enter two positive integers (smaller number first): **5 6**
Double the sum of every second integer in between these values is: 0

```
double num1; // first number input by the user
double num2; // second number input by the user
double sum; // sum of the values
double value; // used for the calculations

// get inputs from user
cout << "Enter two positive integers (smaller number first): ";
```

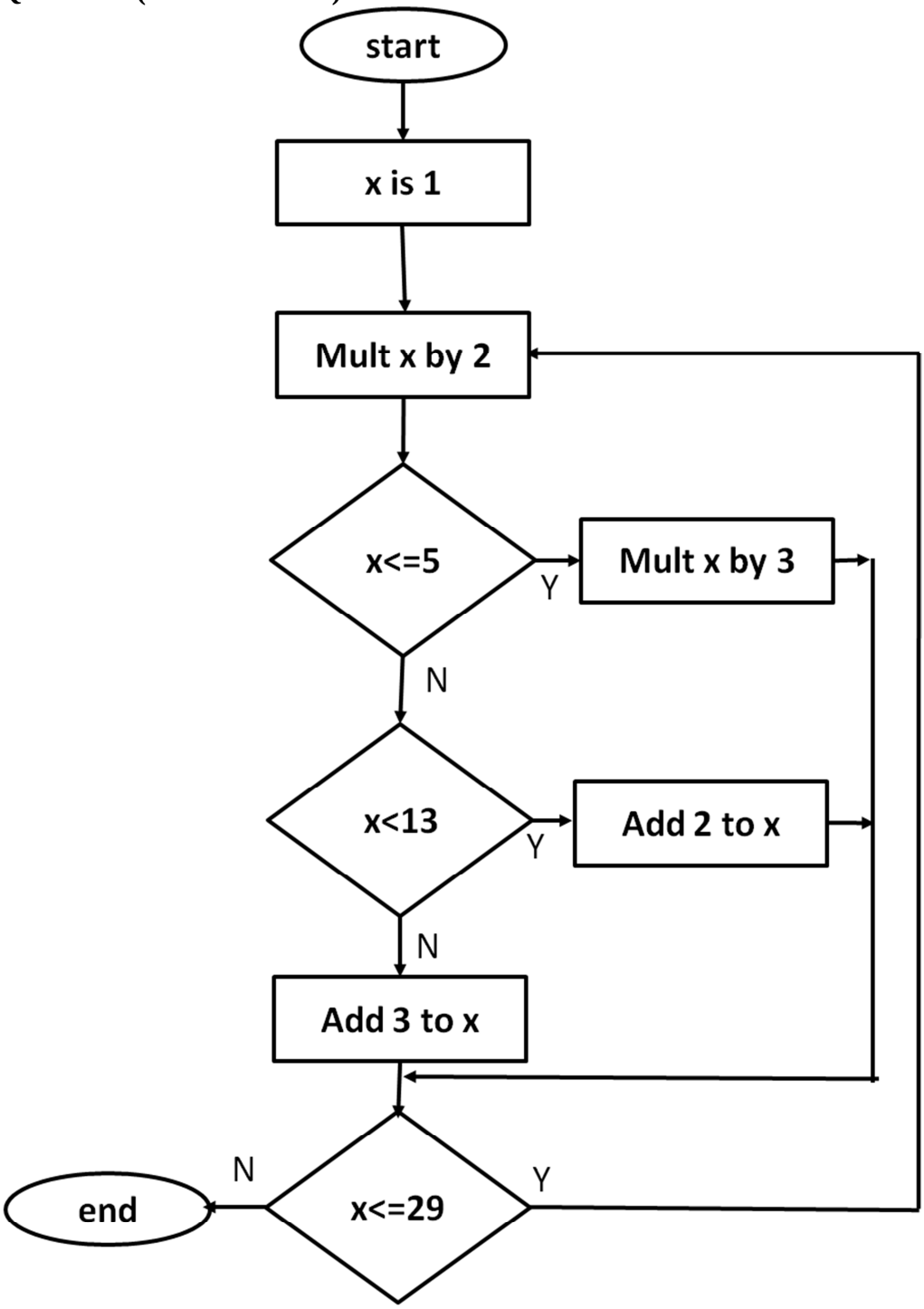
```
    cin >> num1 >> num2;

    // num1 and num2 must be integers, positive, and small<large
    if (!isInt(num1) || !isInt(num2) || num1 <= 0 || num2 <= 0 || num1 >= num2) {
       // invalid input so output error message
       cout << "You did not follow the instructions!" << endl;

    } else {
       // otherwise do the calculation
       sum = 0; // this can also be done in the declarations
       value = num1 + 1; // this variable isn't necessary but makes things a bit clearer

       // loop while value is less than num2, adding 1 each time
       while (value < num2) {
          sum = sum + 2 * value; // add twice value to sum (or sum = sum + value,
                                 // and multiply by 2 at the end)
          value = value + 2; // and increment value
       } // endwhile

       // output results (endl optional); if calculated the sum (not 2 x sum), multiply by 2 here
       // either in the cout or with an additional assignment statement before the cout
       cout << "Double the sum of every second integer in between these values is: "
            << sum << endl;
    } // endif
```

**Marking Scheme:**

**Declarations: 1.5 marks (all variables must be doubles)**
**Get Inputs: 2 marks**
**(1 mark cout; 1 mark cin)**
**Invalid inputs: 2.5 marks**
**(2 for all the error cases; 0.5 for error message)**
**Valid inputs: 3 marks**
**(0.5 initialize sum; 1 while loop; 1 increment sum; 0.5 increment small)**
**Output results: 1 mark**
**Additional deductions / notes:**

- **Deduct 0.5-1 marks if no comments or no useful comments**
- **Deduct 1-2 marks if C--/C++ statements not on page 11 (e.g. for loop, break, etc.) are used**
- **Deduct 1-3 marks for poor indentation, missing/improper ; and { }s**
- **Don't deduct marks if students have used extra variables (and things are otherwise correct)**
- **Don't deduct any marks for missing spaces or "endl"s in the output**
- **Don't deduct marks if student has declared more than one variable on a line, e.g. double value, minimum;**

**Question 2 (2 + 5 = 7 marks):**




i) What is the value of *x* after the execution of the above flowchart?

**31**

ii) Write the C--/C++ code that **best** corresponds to the flowchart. Your code must correspond **exactly** to the flowchart. (No variable declarations or comments required.)

```
x = 1; // 0.5 marks
do {
   x=x*2; // 0.5 marks
   if (x <= 5) { // 0.5 marks
      x = x * 3; // 0.5 marks
   } else if (x < 13) { // 0.5 marks
      x = x + 2; // 0.5 marks
   } else { // 0.5 marks
      x = x + 3; // 0.5 marks
   }
} while (x <= 29)  // 1 mark
```

**Question 3 (2 + 2 + 2 = 6 marks):**

What will be output by each of the following program segments in C++? Note that there is a bonus mark for indicating the output in C--, if it is different. Show the output **exactly** as it would be displayed in the input/output window.

(i)
```
double a = 6;
double b = 2;
do {
   a = a - b;
   b = b - 1;
}  while (b>0);
cout << a << " " << b << endl;
```

**2 0 // 1 mark each; must be on same line with no other text**

(ii)
```
double a = 13;
double b = 6;
cout << 7/4*b << " ";
cout << a/4 << endl;
```

**C--: undefined // 1 bonus mark (thus total of 7 marks maximum for this question)**
**C++: 6 3.75 // 1 mark each; must be on same line with no other text**

(iii)
```
double a = 7;
double b = 3;
while (a < 5 || b < 9) {
   a = a + 2;
   b = b + 1;
}
cout << a << " " << b << endl;
```

**19 9 // 1 mark each; must be on same line with no other text**

**Question 4 (3 + 3 + 4 = 10 marks):**

Complete the following C--/C++ statements. No comments or variable declarations required.

i) A temperature between -20 degrees C and -5 degrees C, **including** those two values, is considered acceptable for a freezer. Assume the temperature is stored in a variable called *temp*.

```
if ( temp >= -20 && temp <= -5 ) {  // 1 mark each for >=, &&, <=
    cout << "That is freezer temperature." << endl;
} // end if
```

ii) A professional sports player receives a contract bonus if he/she scores 50 or more goals, or if he/she achieves a points total of more than 90. Assume the goals count is stored in variable *goals* and the points total is stored in variable *points*.

```
if ( goals >= 50 || points > 90 ) { // 1 mark each for >=, ||, >
    cout << "Contract bonus earned" << endl;
} // end if
```

iii) Assume three variables called *a*, *b*, and *c*. Looping is to continue as long as at least one of these variables is 1, and at least one is not zero. (Do NOT make use of C++'s implicit conversions between "int" and "bool" quantities.)

```
while ( (a==1||b==1||c==1) && (a!=0||b!=0||c!=0) ) { // 1 mark for &&; 1.5 for each side
    // do something
    ....
} // end while
```

**Question 5 (12 marks):**

The following program asks the user to enter a number, and outputs whether it is greater than 50 or less than 50. Don't worry about values equal to 50 – see below.

```
double value;
// get a value from the user
cout << "Enter a number: ";
cin >> value;
// output whether the value is greater than or less than 50
if (value>50) {
   cout << "That number is greater than 50." << endl;
} else {
   cout << "That number is less than 50." << endl;
} // end if
```

You are to **re-write** the above program on the next pages with the following modifications: The program should ask the user to repeatedly input a number, or 50 to stop the program (i.e. add a **sentinel loop**). For each number, output whether it is greater than 50 or not (as above). In addition, when the user enters 50, output the number of numbers entered that were less than 50, the smallest number entered, and the average of all the numbers less than 50 entered. Make sure that you never divide by 0. The examples below should give you the idea (user input in **bold**).

**First Example (all statistics are calculated):**
Enter a number (50 to stop): **75**
That number is greater than 50.
Enter a number (50 to stop): **25**
That number is less than 50.
Enter a number (50 to stop): **85**
That number is greater than 50.
Enter a number (50 to stop): **40**
That number is less than 50.
Enter a number (50 to stop): **50**
Number of numbers less than 50 entered: 2
Smallest number entered: 25
Average of the numbers less than ~~or equal to~~ 50 entered: 32.5

**Second Example (not all statistics are calculated):**
Enter a number (50 to stop): **55.5**
That number greater is than 50.
Enter a number (50 to stop): **50**
Number of numbers less than 50 entered: 0
Smallest number entered: 55.5

```
// variables
double value;
double l50count = 0; // or can set to 0 in an assignment before the loop
                // number of <50 values entered
double l50sum = 0; // or can set to 0 in an assignment before the loop
                  // sum of <50 values entered
double count = 0; // need to count all inputs
double min; // minimum value (no initial value will work)

// get a value from the user
cout << "Enter a number (50 to stop): ";
cin >> value;

// sentinel loop: loop until user enters 50
while (value != 50) {

   count = count + 1; // count inputs
   // if it's the first time or we have a smaller value, change min
   // note that this cannot be inside the if above as smallest number may be >50
   if (count==1 || value < min) {
      min = value;
   } // end if; this code may also be below (but not inside) the next if

   // check to see if it's greater or less than 50
   if (value>50) {
      cout << "That number is greater than 50." << endl;
   } else {
      // less than 50, so need to count and sum
      cout << "That number is less than 50." << endl;
      l50count = l50count + 1; // count it
      l50sum = l50sum + value; // add it to the sum
   } // end if

   // get next value from the user
   cout << "Enter a number (50 to stop): ";
   cin >> value;
} // end while

// output statistics
cout << "Number of numbers less than 50 entered: " << l50count << endl;
// the others only make sense after a check
if (count > 0) { // we have a minimum
   cout << "Smallest number entered: " << min << endl;
}
if (l50count > 0) { // or l50count != 0; we have an average
   cout << " Average of the numbers less than 50 entered: " <<
              l50sum/l50count << endl;
} // end if
```

**Marking Scheme:**

**Declarations: 2 marks (all variables must be doubles)**

**Sentinel loop: 4 marks**

**(2 for value!=50 and 2 for cout and cin before the end while)**

**Tracking Statistics: 4 marks**

**(1 mark incrementing counts; 1 mark adding value to sum;**

**2 marks for correctly updating min: deduct 1 if that if is inside the other if)**

**Output statistics: 2 marks**

**(1 mark for if, 1 for cout – endl optional)**

**Additional deductions / notes:**

- **Deduct 0.5-1 marks if no comments or no useful comments**
- **Deduct 1-2 marks if C--/C++ statements not on page 11 (e.g. for loop, break, etc.) are used**
- **Deduct 1-3 marks for poor indentation, missing/improper ; and { }s**
- **Don't deduct marks if students have used extra variables (and things are otherwise correct)**
- **Don't deduct any marks for missing spaces or "endl"s in the output**
- **Don't deduct marks if student has declared more than one variable on a line, e.g. double value, minimum;**

(more space on next page)

**Constant Declarations:**

**const double** *constantName* **=** *expression* **;** // *real number constant comment*

Notes: The "// … comment" is optional.

**Variable Declarations:**

**double** *variableName* **=** *expression* **;** // *real number variable comment*

Notes: The "= *expression*" is optional. If it is not present the variable does not have a defined initial value. The "// … comment" is also optional.

**Assignment Statements:**

*variableName = expression* **;**

**Input Statements:**

**cin** >> *variableName1* >> *variableName2* … **;**

Notes: The input list may include any number of variable names.

**Output Statements:**

**cout** << *item1* << *item2* … **;**

Notes: The output list may include any number of items. Each item can be a message enclosed in double quotes (e.g. "Hello"), an expression (in which case the value of the expression is output) or *endl* (which ends the current output line).

**While Loops:**

```
while ( trueOrFalseExpression ) {
   ……
} // end while
```

Notes: The brackets around the true or false expression are required. The "while" and the closing brace should be aligned (as shown) and all of the statements within the while loop should be indented by some fixed amount. The "// end while" is in fact just a comment that serves to identify the construct that the closing brace belongs to. It is recommended but may be omitted.

**Do While Loops:**

```
do {
   ……
} while ( trueOrFalseExpression );
```

Notes: The brackets around the true or false expression are required. The "do" and the brace before "while" should be aligned (as shown) and all of the statements within the do while loop should be indented by some fixed amount. Do not overlook the semi-colon at the very end.

**If Statements:**

```
if  ( trueOrFalseExpression ) {
   .....
} else if ( trueOrFalseExpression ) {
   .....
} else if ( trueOrFalseExpression ) {
   .....
} else {
   .....
} // end if
```

Notes: The brackets around the true or false expression are required. The "if" and all right braces should be aligned (as shown) and all of the statements within each section of the if statement should be indented by some fixed amount. There may be any number of "else if" sections (including none at all). The "else" section is optional. The "// end if" is in fact just a comment that serves to identify the construct that the closing brace belongs to. It is recommended but may be omitted.