UNIVERSITY OF WATERLOO

# ECE 150 Midterm Test

**Term:** Fall **Year:** 2016

**TOTAL MARKS: 100**

*This test is worth 10% of a student's final grade for ECE 150.*

**Student Name:** _____

**Signature:** _____

**Student Identification Number:** _____

| | |
|---|---|
| Course Abbreviation and Number: | ECE 150 |
| Course Title: | Fundamentals of Programming |
| **Section (please check one):** | ☐ **001 – Paul Ward** |
| | ☐ **002 – Alfred Yu** |
| | ☐ **003 – Scott Chen** |
| Date of Exam: | Thursday October 20, 2016 |
| Exam Period Start time: 8:35 am | End time: 10:15 am |
| Duration of Exam: | 1 hour 40 minutes |
| Number of Exam Pages: (includes cover page) | 16 |

Exam Type: (select one) ☑ Closed Book ☐ Special Materials ☐ Open Book

Materials Allowed: (select one)

☑ No additional materials are allowed.

☐ Materials allowed are listed below:

**General Exam Policies:**

- No electronic devices including no calculators.
- **Other than cin/cout, no library functions are permitted in solutions.**
- State any assumptions clearly and whenever needed.
- No questions are permitted during the test. If you are unsure about something, make an assumption and write it down.
- If you write the test in pen, you may appeal your test score in the event of discrepancy. However, if you write the test in pencil, you may not appeal your test score except for procedural errors related to incorrect totaling of marks from individual questions.
- You are not allowed to leave the examination room in the first 1 hour of the test (*i.e.*, before 9:35 am) nor in the final 10 minutes of the test (*i.e.*, after 10:05 am).
- *hoc quoque transibit*

**Marking Scheme (110 Marks Available; Test will be Scored Out of 100)**
(If your test score is >100, they will be counted as bonus marks)

| Question | Maximum Score | | Total |
|---|---|---|---|
| 1 | 30 | | |
| 2 | 20 | | |
| 3 | 20 | | |
| 4 | 40 | | |
| | **100** | | |

**Question 1: True or False [30 Marks]**

(+3 marks for each **correct answer**, 0 marks for each instance of **no response**,
–1 mark for each **incorrect answer**)

**Please circle TRUE or FALSE as your response**
**** If you do not wish to respond to a question, do not circle any word. ****

(a) The following code snippet has a console output of: `The answer is 5`

```
int param1 = 10;
int param2 = 1;

int outputParam = (param1 + param2) / 2;

cout << "The answer is " << outputParam << endl;
```

**TRUE** **FALSE**

(b) The following code snippet will output the console message "BINGO!" exactly 5 times.

```
int upperBound = 46;

for (int i = 3; i < upperBound; i++) {
  if (!(i % 8)) {
    cout << "BINGO!" << endl;
  }
}
```

**TRUE** **FALSE**

(c) The following two code snippets will produce identical outputs.

```
int initValue = 2;

for (int i = 0; i < 5; i++) {
  initValue *= 4;
  cout << initValue << endl;
}
```

```
int initValue2 = 2;

for (int j = 5; j > 0; j--) {
  initValue2 <<= 2;
  cout << initValue2 << endl;
}
```

**TRUE** **FALSE**

(d) Assuming the input provided by the user is always correct, the following code snippet will lead to occasional out-of-bound array access.

```
const int arraySize = 8;
int array[arraySize] = {35, 117, 6, -9, 100, -7762, 130, 6};
int input;

cin >> input;

for (int i = 0; i <= (input % arraySize); i++) {
  cout << "Element " << (i + 1) << ": " << array[i] << endl;
}
```

**TRUE** **FALSE**

(e) Provided that the integer variable x is stored in memory location 191, the following C++ code snippet and the assembly code are performing an identical task.

```
while(x < 50){
   x += 2;
}
```

```
771  LD   191, R1
772  ADDi R1, 2, R1
773  SUBi R1, 50, R2
774  JLZ  R2, 772
775  SD   R1, 191
```

**TRUE** **FALSE**

(f) Given that an integer 42 is stored in memory location 553, and integer 82 in location 554, the value to be stored in location 555 will be 6 at the end of the routine.

```
121       LD   553, R1
122       LD   554, R2
123       MULi R1, 2, R1
124       SUB  R1, R2, R1
125       JLEZ R1, 128
126       MULi R1, 3, R1
127       SD   R1, 555
128       JMP  130
129       SD   R2, 555
130       ...
```

**TRUE** **FALSE**

(g) Provided that the integer variable x is stored in memory location 667, and y in 668, the following C++ code snippet and the assembly code snippet are performing an identical task.

```
int y = (x >= 600) ? x - 5 : x * 5;
```

```
471  LD   667, R1
472  SUBi R1, 600, R2
473  JGEZ R2, 476
474  MULi R1, 5, R3
475  JMP  477
476  SUBi R1, 5, R3
477  SD   R3, 668
```

**TRUE** **FALSE**

(h) After executing the following code snippet, the value stored in `thisInt` is 0.

```
int thisInt;
int intA = 15;
int intB = 2;
unsigned int thisIntArray[4] = {0, 3080, 3, 6};
thisInt = intA | intB;
for (int i = 0; i < 4; i++) {
   thisInt = thisInt & thisIntArray[i];
}
```

**TRUE** **FALSE**

(i) The following two code snippets will output exactly the same console messages for all cases of `a`

```
if (a == 2) {
     cout << “Fun!”;
}
else if (a == 3) {
     cout << “is Fun!”;
}
else if (a == 4) {
     cout << “C++ is”;
}
else if (a == 5) {
     cout << “C++ is Fun!”;
}
```

```
switch(a){
  case 5:
    cout << “C++ ”;
  case 3:
    cout << “is ”;
  case 2:
    cout << “Fun!”;
    break;
  case 4:
    cout << “C++ is”;
    break;
}
```

**TRUE** **FALSE**

(j) When running the following code in GDB, by using the command “break 6” prior to running the program, and then using the commands “print x” and “print y” GDB will display the values 13 and 8 for x and y, respectively.

```
1  int main(){
2    int x = 3;
3    int y = 5;
4    int z = 7;
5    y = y + x;
6    z = z * 3;
7    x = z - y;
8    cout << "X: " << x << "\nY: " << y << "\nZ: " << z << endl;
9    return 0;
10 }
```

**TRUE** **FALSE**

**Question 2: Multiple Choice [20 Marks]**
(+5 marks for each **correct answer**, 0 marks for each instance of **no response** or **wrong answer**)

**(a) What is the output of the following program, given the input: `001000100011110`**

```
#include <iostream>
using namespace std;
int main() {
  char input[100];
  cin >> input;
  int i = 0;
  int x = 0;
  int y = 0;
  int invalid = 0;
  while (input[i] != 0 && invalid == 0) {
    switch(input[i]) {
    case '0':
      x++;
      break;
    case '1':
      y++;
      break;
    default:
      invalid = 1;
      break;
    }
    ++i;
  }
  if (invalid) {
    cout << "Invalid Input" << endl;
  } else {
    cout << "x: " << x << "; y: " << y << endl;
  }
  return 0;
}
```

**A:** x: 7; y: 6 **B:** x: 6; y: 7 **C:** x: 6; y: 9 **D**: x: 9; y: 6 **E:** Invalid Input

**(b) What is the output of the following program?**

```
#include <iostream>
using namespace std;
int main() {
  int array[10] = {-11, 6, 0, -2, 9, 10, 17, 3, 42, 13};
  int x[3] = {0, 0, 0};
  for (int i = 0; i < 10; ++i) {
    if (array[i] < 0) {
      x[0]++;
    } else if (array[i] <= 10) {
      x[1]++;
    } else {
      x[2]++;
    }
  }
  for (int i = 0; i < 3; ++i) {
    cout << x[i] << " ";
  }
  cout << endl;
  return 0;
}
```

**A:** 3 4 3 **B:** 2 4 4 **C:** 2 5 3 **D:** 2 7 3 **E:** 3 3 4

**(c) What is the output of the following program?**

```
#include <iostream>
using namespace std;
int main() {
  char s[10] = "string";
  char t[10] = "notString";
  for (int i = 0; i < 6; ++i) {
    t[i] = s[(i+2)%6];
  }
  cout << t << endl;
  return 0;
}
```

**A:** ringst **B:** string **C:** stringing **D:** notString **E:** ringsting

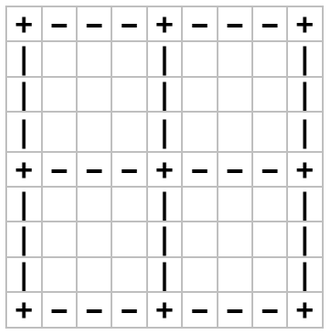
**(d) What is the output of the following program?**

```
#include <iostream>
using namespace std;
int main() {
  char s[100] = "A very long string about a very special ring ...";
  char f[] = "ring";
  int i = 0;
  while (s[i] != 0) {
    int j = 0;
    int m = 1;
    while ((f[j] != 0) && (m == 1)) {
      if (s[i+j] != f[j]) {
        m = 0;
      }
      j++;
    }
    if (m == 1) {
      cout << i << " ";
    }
    ++i;
  }
  return 0;
}
```

**A:** 13 **B:** 14 **C:** 13 39 **D:** 14 40 **E:** 40

**Question 3 [20 Marks]**

Write a C++ code snippet to draw a four-box grid in a command-line console window. Each side of box is of dimension `size`, which is defined in terms of the number of characters (inclusive of the walls). As an example, if `size` is 5, the result should look like the following:



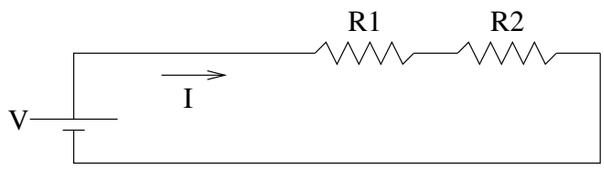
Here are a few drawing rules:

- Horizontal box sides should be depicted using the symbol **–** (minus sign)
- Vertical box sides should be depicted using the symbol **|** (vertical bar sign)
- Each box corner should be depicted using the symbol **+** (plus sign)
- If a particular pixel position should be blank within the grid, output a whitespace character to the console

Your C++ code snippet would be placed within the following program:

```
#include <iostream>
using namespace std;

int main(){
     int size;
     cin >> size;
     if (cin.fail() || size <= 1)
          return -1;

     //
     // YOUR CODE SNIPPET WILL BE PLACED HERE
     //

     return 0;
}
```

**Question 4 [40 Marks]** Consider the following electrical circuit:




While we know that **V = I(R1+R2)** from the ECE 140 course, the accuracy of resistors may readily vary up to ±20% (per ECE 150, Assignment 2, Task 3). Therefore, the value of the current, **I**, would depend on the actual value of the resistors, **R1** and **R2**.

**(a) [5 Marks**] Of concern in electrical circuits is the amount of current flowing through resistors, since they may overheat if the current is too high. Write a C++ code snippet to compute the maximum values of the current, I, given values for:

- Nominal voltage, V (5.2 volts)
- Resistor 1's nominal value, R1 (10000 ohms)
- Error percentage for R1 (±5% deviation)
- Resistor 2's nominal value, R2 (3000 ohms)
- Error percentage for R2 (±20% deviation)

Output to the console the maximum current value you compute.

```
#include <iostream>
using namespace std;

int main(){
  float V, R1, err1, R2, err2;

  V = 5.2;              // Nominal voltage; subject to change
  R1 = 10000;           // Nominal resistance; subject to change
  err1 = 5;             // Resistor 1's deviation (in %)
  R2 = 3000;            // Nominal resistance; subject to change
  err2 = 20;            // Resistor 2's deviation (in %)


   //
   // YOUR CODE SNIPPET WILL BE PLACED HERE
   // Don't forget to output the values using cout
   //

  return 0;
}
```

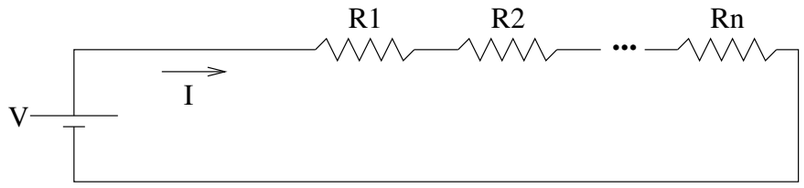
**(b) [5 Marks]** Also of concern in electrical circuits is the voltage level over the individual resistors, since the resistor may breakdown if the voltage across it is too high. Write a C++ code snippet to compute the maximum values of the voltage over each of the two resistors, R1 and R2, given values for the voltage, V, the resistors, R1 and R2, together with the error percentage for R1 and R2. Output to the console the maximum voltage values you compute.

Hint 1: The **maximum voltage over R1** would correspond to the scenario where the error in R1 makes R1 equal to its maximum resistance and the error in R2 makes it equal to its minimum resistance.

Hint 2: On the other hand, the **maximum voltage over R2** would correspond to the scenario where the error in R1 makes R1 equal to its minimum resistance and the error in R2 makes it equal to its maximum resistance.

```
#include <iostream>
using namespace std;

int main(){
  float V, R1, err1, R2, err2;

  V = 5.2;              // Nominal voltage; subject to change
  R1 = 10000;           // Nominal resistance; subject to change
  err1 = 5;             // Resistor 1's deviation (in %)
  R2 = 3000;            // Nominal resistance; subject to change
  err2 = 20;            // Resistor 2's deviation (in %)


    //
    // YOUR CODE SNIPPET WILL BE PLACED HERE
    // Don't forget to output the values using cout
    //

  return 0;
}
```

**(c) [20 Marks]** Now consider a circuit with a series of resistors, each with its own error percentage:




**(i) [10 marks]** Write a C++ code snippet to compute the maximum value of the current, I, given values for:

- Nominal voltage (V)
- Total number or resistors (N)
- A data array of resistor values (R1, R2, …RN)
- A data array of error values for each resistor (err1, err2, …, errN)

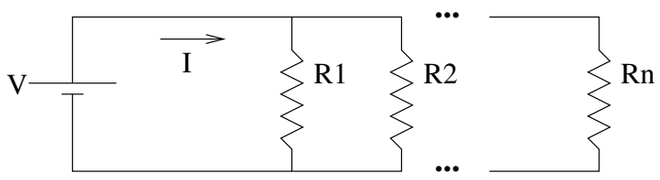
```
#include <iostream>
using namespace std;

int main(){

  // Data initialization for a case with 3 resistors
  const int N = 3;                  // Total number of resistors
  float V = 5.2;                    // Nominal voltage
  float R[N] = {100, 20000, 5000}; // Resistor data array
  float err[N] = {5, 0.25, 20};     // Error data array

   //
   // YOUR CODE SNIPPET WILL BE PLACED HERE
   // Your snippet should work with
   // Don't forget to output the values using cout
   //

  return 0;
}
```

**(ii) [10 marks]** Write a C++ code snippet to compute and output the maximum voltage possible across each resistor.

Hint: The **maximum voltage over Rn** would correspond to the scenario where the error in Rn makes Rn equal to its maximum resistance, while all other resistors are operating at their minimum resistance.

```
#include <iostream>
using namespace std;

int main(){

  // Data initialization for a case with 3 resistors
  const int N = 3;                  // Total number of resistors
  float V = 5.2;                    // Nominal voltage
  float R[N] = {100, 20000, 5000}; // Resistor data array
  float err[N] = {5, 0.25, 20};     // Error data array

  //
  // YOUR CODE SNIPPET WILL BE PLACED HERE
  // Don't forget to output the values using cout
  //

  return 0;
}
```

**(d) [10 Marks]** Finally, consider a circuit with a collection of resistors connected in parallel, each with error percentages:




Write a C++ code snippet to compute the maximum value of each branch current, Ii, through each resistor Ri, when given values for:

- Nominal voltage (V)
- Total number or resistors (N)
- A data array of resistor values (R1, R2, …RN)
- A data array of error values for each resistor (err1, err2, …, errN)

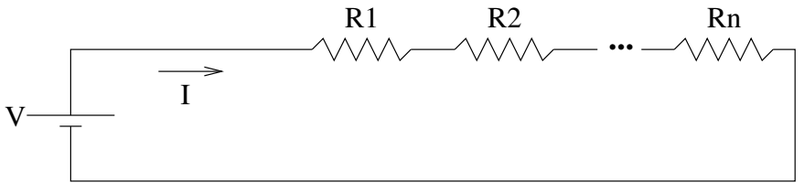
Hint: The voltage is the same over all the resistors R1 to Rn.

```
#include <iostream>
using namespace std;

int main(){

  // Data initialization for a case with 3 resistors
  const int N = 3;                  // Total number of resistors
  float V = 5.2;                    // Nominal voltage
  float R[N] = {100, 20000, 5000}; // Resistor data array
  float err[N] = {5, 0.25, 20};     // Error data array

    //
    // YOUR CODE SNIPPET WILL BE PLACED HERE
    // Don't forget to output the values using cout
    //

  return 0;
}
```

# RISC Machine-Level (Assembly) Instructions

```
PC: Program Counter
Ri: Register i for any particular i
nnnn: Numerical value

LD <nnnn>, Ri       Ri <-- data at memory location <nnnn>
LD Ri, Rj           Rj <-- data at memory location contained in Rj
LDi nnnn, Ri        Ri <-- nnnn
SD Ri, <nnnn>       Store data in Ri to memory location <nnnn>
SD Ri, Rj           Store data in Ri to memory location contained in Rj

ADD Ri, Rj, Rk      Rk <-- Ri + Rj
ADDi Ri, nnnn, Rj   Rj <-- Ri + nnnn
SUB Ri, Rj, Rk      Rk <-- Ri - Rj
SUBi Ri, nnnn, Rj   Rj <-- Ri - nnnn
MUL Ri, Rj, Rk      Rk <-- Ri * Rj
MULi Ri, nnnn, Rj   Rj <-- Ri * nnnn
DIV Ri, Rj, Rk      Rk <-- Ri / Rj
DIVi Ri, nnnn, Rj   Rj <-- Ri / nnnn

JMP <nnnn>          PC <-- nnnn
JZ Ri, <nnnn>       PC <-- nnnn if Ri == 0
JNZ Ri, <nnnn>      PC <-- nnnn if Ri != 0
JGZ Ri, <nnnn>      PC <-- nnnn if Ri > 0
JGEZ Ri, <nnnn>     PC <-- nnnn if Ri >= 0
JLZ Ri, <nnnn>      PC <-- nnnn if Ri < 0
JLEZ Ri, <nnnn>     PC <-- nnnn if Ri <= 0
```

# Series Electrical Circuits




Given a series electrical circuit, as shown above, the following equations describe it:

(1) The current, I, is the same through all resistors, R1, R2, … Rn. It is:

$$I = V / (R1 + R2 + \ldots Rn)$$

(2) The voltage, Vi, over each resistor, Ri, is determined by the current flowing through the resistor:

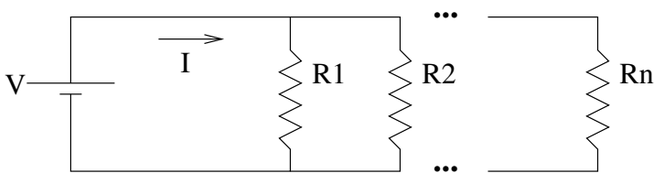
$$Vi = I * Ri$$

*E.g.*, $V1 = I * R1$

(3) Given (1) and (2) above, equation (2) may be re-written as:

$$Vi = V * Ri / (R1 + R2 + \ldots Rn)$$

# Parallel Electrical Circuits




Given a parallel electrical circuit, as shown above, the following equations describe it:

(1) The voltage, V, is the same over all resistors, R1, R2, … Rn.

(2) The current, Ii, through each resistor, Ri, is then:

$$Ii = V / Ri$$

*E.g.*, $I1 = V / R1$

# C++ Operators

```
x++         add 1 to x; return x before the addition
++x         add 1 to x; return x after the addition
x--         subtract 1 from x; return x before the subtraction
--x         subtract 1 from x; return x after the subtraction
x * y       return x multiplied by y
x / y       return x divided by y
            (integer division if both x and y are integers)
x + y       return x plus y
x - y       return x - y
x % y       return the remainder of x divided by y
cin >> x    take input from user and store it in x
cout << y   print value of y to the user
x && y      logical AND of x and y
x || y      logical OR of x and y
!x          logical NEGATION of x
            (if a variable is 0, it is treated a ''false'')
            (if it is anything other than 0, it is ''true'')
            (if x is 0, !x is 1)
x & y       bitwise AND of x and y
x | y       bitwise OR of x and y
x ^ y       bitwise EXCLUSIVE OR of x and y
~x          bitwise NEGATION of x
x >> y      right bitshift x by y bits
x << y      left bitshift x by y bits
            (x should be unsigned or the results can be undefined)
x ? y : z   if x is true, evaluate y and return the result;
            else evaluate z and return the result
```