CPE 150
INTRODUCTION TO PROGRAMMING
FIRST EXAM

Department of Computer Engineering
Yarmouk University
July 22, 2017

This is a CLOSED BOOK exam. Textbooks, notes, laptops, calculators, personal digital assistants, cell phones, and Internet access are NOT allowed.

It is a 60 minute exam, with a total of 15 marks. There are 2 sections, 8 questions, and 7 pages (including this cover page). Please read each question carefully, and write your answers legibly in the space provided. You may do the questions in any order you wish, but please USE YOUR TIME WISELY.

When you are finished, please hand in your exam paper and sign out. Good luck!

Name: _____

Student I.D.: _____

Instructor and Section: _____

Section 1: Program Comprehension and Debugging (7.5 marks)

Q1. (*1.5 mark*) Consider the code in listing 1: How many times will the statement at line **2** execute if n is defined as an `int` and initialized to **10**? and why?

```
1 while (n > 0)
2     n /= 2;
```

Listing 1: Code for Q1

Answer for Q1:

(0.5 mark) *How many times line* 2 *is executed?*

(1 mark) *Explain why.*

Q2. (*1 mark*) Consider the code in listing 2: What does the code do? Write a mathematical expression that defines the relation between `result`, `val1` and `val2` after executing the code. [*Note*: assume that `val1` and `val2` are greater than 0]

```
1 int val1, val2, result = 1;
2 cout << "Enter two positive integer numbers: ";
3 cin >> val1 >> val2;
4 while(val2 > 0){
5     result *= val1;
6     val2--;
7 }
```

Listing 2: Code for Q2

Answer for Q2:

Q3. (*1 mark*) Consider the code in listing 3: What does the code do to the values of the variables `i` and `j`. [*Note*: the answer should be about **what** the code do not **how** its done]

```
1 int i, j;
2 cout << "Enter two integer numbers: ";
3 cin >> i >> j;
4 i = i + j;
5 j = i - j;
6 i = i - j;
```

Listing 3: Code for Q3

Answer for Q3:

Q4. (*2 marks*) Rewrite the code in listing 4 so that it uses a `do/while` loop construct to do the same task.

```
1  int n;
2  cout << "Enter a non-negative integer: ";
3  cin >> n;
4  while (n < 0)
5  {
6      cout << "The integer you entered is negative." << endl;
7      cout << "Enter a non-negative integer: ";
8      cin >> n;
9  }
```

Listing 4: Code for Q4

Answer for Q4:

Q5. (*1 mark*) Consider the code in listing 5: What does the code do? Write a mathematical expression that defines the relation between `result`, `a` and `b` after executing the code. [*Note*: assume that `a` and `b` are greater than or equal to 0]

```
1  int a, b, result = 0;
2  cout << "Enter two positive integers: ";
3  cin >> a >> b;
4  while(b != 0)
5  {
6      result += a;
7      b--;
8  }
```

Listing 5: Code for Q5

Answer for Q5:

Q6. (*1 mark*) The code in listing 6 is meant to produce the triangle of stars in listing 8, however, running the code in listing 6 produces the rectangle of stars in listing 7. What is the **single change** that you can make so the code in listing 6 generates the triangle of stars in listing 8?

[*Note*: You are permitted a single change of an operator, conditional values, or initializations. You cannot add new constructs to the code.]

```
1 for(int i = 0; i <= 6; i++)
2 {
3     for(int j = 6; j >= 0; j--)
4     {
5         cout << "*";
6     }
7     cout << endl;
8 }
```

Listing 6: Errorneous code for Q6

```
**********
**********
**********
**********
**********
**********
**********
```

Listing 7: Errorneous output by running the code in listing 6

```
1 *******
2 ******
3 *****
4 ****
5 ***
6 **
7 *
```

Listing 8: Intended output for Q6

Answer for Q6:

Section 2: Programming Skills (7.5 marks)

Q1. (*3.5 marks*) Write a C++ program that reads two integer values from the user x and y. Then, round the value of x to the nearest multiple of y not greater than x. Listing 9 shows two sample runs of the intended program.

[*Note*: assume that $x > y$ and $x, y \neq 0$]

```
Sample Run 1:
Enter x: 25
Enter y: 6
Result: 25 is rounded to 24

Sample Run 2:
Enter x: 52
Enter y: 7
Result: 52 is rounded to 49
```

Listing 9: Sample Output for Q1

Answer for Q1:

```
int main() {









    return 0;
}
```

Q2. (*4 marks*) An odd number is an integer number which can not be divided by 2. Write a C++ program to find the **sum of odd digits** present in a given number. Your program must ask the user for an integer number then output the sum of odd digits in that number. Listing 10 shows two sample runs of the intended program. You **must** use loop constructs to handle any number of digits. *Handling fixed number of digits will award you zero.*

[*Note*: assume that integer number enetered by the user is always greater than zero]

```
Sample Run 1:
Enter an integer number: 256
The sum of odd digits present in 256 is: 5

Sample Run 2:
Enter an integer number: 2734
The sum of odd digits present in 2734 is: 10
```

Listing 10: Sample Output for Q2

Answer for Q2:

```
int main() {









    return 0;
}
```

C++ Data Types

Data Type	Description
`char`	Character
`unsigned char`	Unsigned Character
`int`	Integer
`short int`	Short integer
`short`	Same as `short int`
`unsigned short int`	Unsigned short integer
`unsigned short`	Same as `unsigned short int`
`unsigned int`	Unsigned integer
`unsigned`	Same as `unsigned int`
`long int`	Long integer
`long`	Same as `long int`
`unsigned long int`	Unsigned long integer
`unsigned long`	Same as `unsigned long int`
`float`	Single precision floating point
`double`	double precision floating point
`long double`	Long double precision floating point

Forms of the `if` Statement

Simple if

```
if (expression)
   statement;
```

Example

```
if (x < y)
   x++;
```

if/else

```
if (expression)
   statement;
else
   statement;
```

Example

```
if (x < y)
   x++;
else
   x--;
```

if/else if

```
if (expression)
   statement;
else if (expression)
   statement;
else
   statement;
```

Example

```
if (x < y)
   x++;
else if (x < z)
   x--;
else
   y++;
```

To conditionally-execute more than one statement, enclose the statements in braces:

Form

```
if (expression)
{
   statement;
   statement;
}
```

Example

```
if (x < y)
{
   x++;
   cout << x;
}
```

Commonly Used Operators

Assignment Operators

`=`	Assignment
`+=`	Combined addition/assignment
`-=`	Combined subtraction/assignment
`*=`	Combined multiplication/assignment
`/=`	Combined division/assignment
`%=`	Combined modulus/assignment

Arithmetic Operators

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`%`	Modulus (remainder)

Relational Operators

`<`	Less than
`<=`	Less than or equal to
`>`	Greater than
`>=`	Greater than or equal to
`==`	Equal to
`!=`	Not equal to

Logical Operators

`&&`	AND
`||`	OR
`!`	NOT

Increment/Decrement

`++`	Increment
`--`	Decrement

Conditional Operator ?:

Form:

```
expression ? expression : expression
```

Example:

```
x = a < b ? a : b;
```

The statement above works like:

```
if (a < b)
   x = a;
else
   x = b;
```

The `while` Loop

Form:

```
while (expression)
   statement;

while (expression)
{
   statement;
   statement;
}
```

Example:

```
while (x < 100)
   cout << x++ << endl;

while (x < 100)
{
   cout << x << endl;
   x++;
}
```

The do-`while` Loop

Form:

```
do
   statement;
while (expression);

do
{
   statement;
   statement;
} while (expression);
```

Example:

```
do
   cout << x++ << endl;
while (x < 100);

do
{
   cout << x << endl;
   x++;
} while (x < 100);
```

The `for` Loop

Form:

```
for (initialization; test; update)
   statement;

for (initialization; test; update)
{
   statement;
   statement;
}
```

Example:

```
for (count = 0; count < 10; count++)
   cout << count << endl;

for (count = 0; count < 10; count++)
{
   cout << "The value of count is ";
   cout << count << endl;
}
```

The `switch/case` Construct

Form:

```
switch (integer-expression)
{
   case integer-constant:
      statement(s);
      break;
   case integer-constant:
      statement(s);
      break;
   default :
      statement;
}
```

Example:

```
switch (choice)
{
   case 0 :
      cout << "You selected 0.\n";
      break;
   case 1 :
      cout << "You selected 1.\n";
      break;
   default :
      cout << "You did not select 0 or 1.\n";
}
```

Using `cout`

Requires <iostream> header file.

Commonly used stream manipulators

Name	Description
`endl`	advances output to the beginning of the next line.
`fixed`	sets fixed point notation
`left`	sets left justification
`right`	sets right justification
`setprecision`	sets the number of significant digits
`setw`	sets field width
`showpoint`	forces decimal point & trailing zeros to display

Example:

```
cout << setprecision(2) << fixed
     << left << x << endl;
```

Member functions for output formatting

Name	Description
`.precision`	sets the number of significant digits
`.setf`	sets one or more `ios` flags
`.unsetf`	clears one or more `ios` flags
`.width`	sets field width

Example:

```
cout.precision(2);
```

Using `cin`

Requires <iostream> header file

Commonly used stream manipulators

Name	Description
`setw`	sets field width

Member functions for specialized input

Name	Description
`.getline`	reads a line of input as a C-string
`.get`	reads a character
`.ignore`	ignores the last character entered
`.width`	sets field width