CPE 150
INTRODUCTION TO PROGRAMMING
SECOND EXAM

Department of Computer Engineering
Yarmouk University
August 5, 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, 10 questions, and 9 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 (8.5 marks)

Q1. (*1 mark*) Show the expected output for the code in listing 1.

```
1  void foo(int [], int);
2  const int size = 5;
3  int main()
4  {
5      int x[size] = {1, 2, 3, 4, 5};
6      foo(x, 2);
7      foo(x, -2);
8      for(int i = 0; i < size; i++)
9          cout << x[i] << " ";
10     cout << endl;
11     return 0;
12 }
13
14 void foo(int arr[], int f)
15 {
16     for(int i = 0; i < size; i++)
17         arr[i] *= f;
18 }
```

Listing 1: Code for Q1

Output for code in listing 1:

Q2. (*1 mark*) Show the expected output for the code in listing 2.

```
1  void someFunction(const int a[], const int size)
2  {
3      for(int i = size - 2 ; i >= 1; i--)
4          cout << a[i] << " ";
5  }
6
7  int main()
8  {
9     const int arraySize = 10;
10    int a[arraySize] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
11    cout << "The values in the array are: " << endl;
12    someFunction(a, arraySize);
13    cout << endl;
14    return 0;
15 }
```

Listing 2: Code for Q2

Output for code in listing 2:

Q3. (*1.5 mark*) Show the expected output for the code in listing 3.

```
1  int mystery(int, int);
2  int main()
3  {
4     int x = 5, y = 4;
5     cout << "The result is " << mystery(x, y) << endl;
6     return 0;
7  }
8
9  int mystery(int a, int b)
10 {
11    if (b == 1)
12       return a;
13    else
14       return a + mystery(a, b - 1);
15 }
```

Listing 3: Code for Q3

Output for code in listing 3:

Q4. (*1 mark*) Show the expected output for the code in listing 4.

```
1  int bar(int);
2  int main()
3  {
4      cout << "Result is " << bar(5) << endl;
5      cout << "Result is " << bar(10) << endl;
6      return 0;
7  }
8
9  int bar(int x)
10 {
11     static int y = 20;
12     int swap = y;
13     y = x;
14     x = swap;
15     return x;
16 }
```

Listing 4: Code for Q4

Output for code in listing 4:

Q5. (*1.5 mark*) Show the expected output for the code in listing 5.

```
1  void withDef(int hisNum = 30);
2  void control(int);
3  int main()
4  {
5      int yourNum = 20;
6      control(yourNum);
7      withDef();
8      cout << "Number = " << yourNum << endl;
9      return 0;
10 }
11
12 void withDef(int hisNum)
13 {
14     for(int i = 20; i <= hisNum; i+=5)
15         cout << i << " ";
16     cout << endl;
17 }
18
19 void control(int myNum)
20 {
21     myNum += 10;
22     withDef(myNum);
23 }
```

Listing 5: Code for Q5

Output for code in listing 5:

Q6. (*1 mark*) Show the expected output for the code in listing 6.

```
1  void position(int &, int);
2  int c2 = 10;
3  int p2 = 20;
4  int main()
5  {
6      int p1 = 20, p2 = 4;
7      position(p1, 3);
8      cout << p1 << ", " << p2 << endl;
9      position(p2, p1);
10     cout << p1 << ", " << p2 << endl;
11     return 0;
12 }
13
14 void position(int &c1, int c2)
15 {
16     c1 += 2;
17     c2 += 1;
18 }
```

Listing 6: Code for Q6

Output for code in listing 6:

Q7. (*1.5 mark*) Show the expected output for the code in listing 7. Suppose that in response to the first `cin` the user types the following line and presses `Enter`:

```
Please go away.
```

```
1  int main()
2  {
3      const int LENGTH = 9;
4      char message[LENGTH];
5      cout << "Enter a sentence on the line below." << endl;
6      int i = 0;
7      do
8      {
9          cin >> message[i];
10         ++i;
11     }
12     while(i < LENGTH - 1 && message[i] != '\n');
13     message[i] = '\0';
14     cout << "[" << message << "]" << endl;
15     return 0;
16 }
```

Listing 7: Code for Q7

Output for code in listing 7:

Section 2: Programming Skills (8.5 marks)

Q1. (*3 marks*) Complete the code below in listing 8 that rolls a dice 10000 and uses `rand` function to generate random faces $(1, 2, 3, 4, 5, 6)$ for the dice each roll, then prints the frequency for each face as in the sample output in listing 9.

```
int main()
{
    const int arraySize = 7;
    // An array to store the frequency of each face when drawing the dice.
    // frequency of face 1 is stored at index 1, face 2 at index 2 and so on.
    int frequency[arraySize] = {0};
    srand(time(0));  // seed random-number generator

    // roll die 10000 times
    for ( int roll = 1; _____; roll++ )
    {
        // Call rand() function with proper scaling and shifting
        // to generate random numbers between 1 and 6 that represent a die face.

        int face = _____;

        // Increase the frequency of face in frequency array by 1.

        _____;
    }

   cout << "Face    Frequency" << endl;
   // output frequency elements 1-6 in tabular format

   for ( int face = _____; face < arraySize; face++)
   {
       // print the face and its frequency.

       cout << face << "       " << _____ << endl;
   }
   return 0;
}
```

Listing 8: Code for Q1

```
Face    Frequency
1       1691
2       1641
3       1707
4       1674
5       1606
6       1681
```

Listing 9: Sample Output for Q1

Q2. (*3 marks*) Complete the function `printHistogram` in listing 10 so that it prints the histogram shown on the sample output in listing 11. Assume that the values in the array `A` are in the range from 0 to 9 only.

```
void printHistogram(int A[], int size){
    // Freq stores the number of times each number in array A is repeated,
    // it must be initialized to zero for all elements.

    int Freq[10] = _____________________________;

    // loop through all elements in array A to store frequencies in Freq.
    for(int i = 0; i < size; i++)
    {
        // increment the frequency for the number A[i] in array Freq by 1

        _____________________________;
    }
    cout << "Value        Histogram" << endl;
    // loop through all elements in array Freq to print histogram
    for(int i = 0; i < 10; i++)
    {
        cout << i << "            ";

        if(_____________________________)
        {
            cout << "* (Unique)";

        }else if(_____________________________)
        {
            cout << "(Zero)";
        }else{
            // print stars according to the frequency

            for(int j = 0; _____________________________ ; j++)
                cout << "*";
        }
        cout << endl;
    }
}
void main() {
   const int arraySize = 20;
   int arr[arraySize] = {1,1,1,2,2,2,2,2,2,3,3,0,6,6,8,7,6,0,9,9};
   printHistogram(arr, arraySize);
}
```

Listing 10: Code for Q2

```
Value        Histogram
0            **
1            ***
2            ******
3            **
4            (Zero)
5            (Zero)
6            ***
7            * (Unique)
8            * (Unique)
9            **
```

Listing 11: Sample Output for Q2

Q3. (*2.5 marks*) Complete the `f_iterative` function in listing 12 to match the functionality performed in the recursive function `f_recursive`.

```
int f_iterative(int x){
    // x0 is the term f(0).
    // x1 is the term f(1).
    // x2 is the term f(2).
    int x0 = 1, x1 = 1, x2 = 2, result;
    if(x <= 0)
            return x0;
    if(x == 1)
            return x1;
    if(x == 2)

            ________________;

    for(int i = 3; ________________  ; i++)
    {
         // set the result from the previous values according to:
         // f(x) = f(x-1) + f(x-2) * f(x-3)

         result = ____________________________________ ;

         // update x0, x1, and x2 to proper new values.

         x0 = ________________ ;

         x1 = ________________ ;
         x2 = result ;
    }
    return result;
}
int f_recursive(int x)
{
    // base case for x = 0
    if(x <= 0)
        return 1;

    // base case for x = 1 and x = 2
    if(x == 1 || x == 2)
        return x;

    // recursive step: f(x) = f(x-1) + f(x-2) * f(x-3)
    return f_recursive(x - 1) + f_recursive(x - 2) * f_recursive(x- 3);
}
void main(){
    cout << "Enter number: ";
    int x;
    cin >> x;
    cout << "Result via recursive is: " << f_recursive(x) << endl;
    cout << "Result via iterative is: " << f_iterative(x) << endl;
}
```

Listing 12: Code for Q3

```
Enter number: 5
Result via recursive is: 11
Result via iterative is: 11
```

Listing 13: Sample Output for Q3

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