CPE 150
INTRODUCTION TO PROGRAMMING
FINAL EXAM

Department of Computer Engineering
Yarmouk University
August 21, 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 120 minute exam, with a total of 35 marks. There are 2 sections, 14 questions, and 11 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 (25 marks)

Q1. (*2.5 marks*) Show the expected output for the code in listing 1.

```
1  const int rows = 2;
2  const int cols = 3;
3  void foo(int arr[], int f)
4  {
5      for(int i = 0; i < cols; i++)
6          arr[i] *= f;
7  }
8  int main()
9  {
10     cout << "*** Output 1 ***" << endl;
11     int x[rows][cols] = {{1, 2, 3}, {4, 5, 6}};
12     foo(x[0], 2);
13     foo(x[1], -2);
14     for(int row = 0; row < rows; row++){
15         for(int col = 0; col < cols; col++)
16             cout << x[row][col] << " ";
17         cout << endl;
18     }
19     return 0;
20 }
```

Listing 1: Code for Q1

Output for code in listing 1:

Q2. (*2.5 marks*) 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 - 3 ; i >= 2; i--)
4          cout << a[i] << " ";
5  }
6
7  int main()
8  {
9      cout << "*** Output 2 ***" << endl;
10     const int arraySize = 10;
11     int a[arraySize] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
12     cout << "The values in the array are: " << endl;
13     someFunction(a, arraySize);
14     return 0;
15 }
```

Listing 2: Code for Q2

Output for code in listing 2:

Q3. (*2.5 marks*) Show the expected output for the code in listing 3.

```
1  void someFunction(int b[], int current, int size)
2  {
3      if (current < size){
4          someFunction(b, current + 2, size);
5          cout << b[current] << "  ";
6      }
7  }
8
9  int main()
10 {
11     cout << "*** Output 3 ***" << endl;
12     const int arraySize = 10;
13     int a[arraySize] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
14     cout << "The values in the array are:" << endl;
15     someFunction(a, 0, arraySize);
16     return 0;
17 }
```

Listing 3: Code for Q3

Output for code in listing 3:

Q4. (*2.5 marks*) Show the expected output for the code in listing 4.

```
1  int bar(int x)
2  {
3      static int y = 30;
4      int swap = y;
5      y = x;
6      x = swap;
7      return swap;
8  }
9
10 int main()
11 {
12     cout << "*** Output 4 ***" << endl;
13     cout << "Result is " << bar(6) << endl;
14     cout << "Result is " << bar(17) << endl;
15     return 0;
16 }
```

Listing 4: Code for Q4

Output for code in listing 4:

Q5. (*2.5 marks*) Show the expected output for the code in listing 5.

```
1  void foo(int *p1, int *p2)
2  {
3      ++(*p1);
4      p1++;
5      p1 = p1 - 1;
6      (*p2)--;
7  }
8
9  int main()
10 {
11     cout << "*** Output 5 ***" << endl;
12     int x = 2, y = 3;
13     foo(&x, &y);
14     cout << x << endl;
15     cout << y << endl;
16     return 0;
17 }
```

Listing 5: Code for Q5

Output for code in listing 5:

Q6. (*2.5 marks*) Show the expected output for the code in listing 6.

```
1  void position(int c1, int &c2)
2  {
3      c1 += 3;
4      c2 -= 3;
5  }
6
7  int main()
8  {
9      cout << "*** Output 6 ***" << endl;
10     int p1 = 19, p2 = 5;
11     position(3, p1);
12     cout << p1 << ", " << p2 << endl;
13     position(p2, p1);
14     cout << p1 << ", " << p2 << endl;
15     return 0;
16 }
```

Listing 6: Code for Q6

Output for code in listing 6:

Q7. (*3.5 marks*) Show the expected output for the code in listing 7. Suppose that in response to the first `cin.get` call the user types the following line and presses `Enter`:

```
Please go away.
```

```
1  int main()
2  {
3      cout << "*** Output 7 ***" << endl;
4      const int LENGTH = 12;
5      char message[LENGTH];
6      cout << "Enter a sentence on the line below." << endl;
7      int i = 0;
8      do {
9          cin.get(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:

Q8. (*2.5 mark*) Show the expected output for the code in listing 8.

```
1  int mystery(char *s1, char *s2) {
2      for ( ; *s1 != '\0' && *s2 != '\0'; s1++, s2++)
3      {
4          if (*s1 != *s2)
5              return 0;
6      }
7      return 1;
8  }
9  int main()
10 {
11     cout << "*** Output 8 ***" << endl;
12     char string1[80] = "hi there buddy";
13     char *string2 = "hi there";
14     cout << "The result is " << mystery(string1, string2) << endl;
15     *(string1 + 2) = '\0';
16     cout << "[" << string1 << "]" << endl;
17     return 0;
18 }
```

Listing 8: Code for Q8

Output for code in listing 8:

Q9. (*2.5 mark*) Show the expected output for the code in listing 9.

```
1  int mystery(char *s)
2  {
3      int x = 0;
4      for (; *s != '\0'; s++)
5          ++x;
6      return x;
7  }
8
9  int main()
10 {
11     cout << "*** Output 9 ***" << endl;
12     char string1[80] = "exam is easy";
13     cout << mystery(string1) << endl;
14     string1[4] = '\0';
15     cout << mystery(string1) << endl;
16     return 0;
17 }
```

Listing 9: Code for Q9

Output for code in listing 9:

Q10. (*1.5 marks*) Show the expected output for the code in listing 10.

```
1  void mystery(int *ptr, int size)
2  {
3      int swap = ptr[0];
4      ptr[0] = *(ptr + (size - 1));
5      *(ptr + (size - 1)) = swap;
6      *(ptr + (size / 2)) -= 2;
7  }
8
9  int main()
10 {
11     cout << "*** Output 10 ***" << endl;
12     const int size = 4;
13     int arr[size] = {1, 2, 3, 4};
14     mystery(arr, size);
15     for(int i = 0; i < size; i++)
16         cout << *(arr + i) << " ";
17     cout << endl;
18     return 0;
19 }
```

Listing 10: Code for Q10

Output for code in listing 10:

Section 2: Programming Skills (15 marks)

Q1. (*4 marks*) Complete the code in listing 11 that rolls a dice 10000 times 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 12.

```
int main()
{
    // Define an integer array 'freqArr' of '7' elements and set all elements to zero.
    // We will use 'freqArr' 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.

    ____________________________________________________;

    srand(time(0));  // seed random-number generator

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

        int dieFace = _____________________________;

        // Increase the frequency of 'dieFace' in 'freqArr' array by 1.

        ____________________________________________________;
    }

    cout << "Face    Frequency" << endl;

    // output frequency elements 1-6 in tabular format
    for ( int dieFace = 1; dieFace < 7; dieFace++)
    {
        // print the face and its frequency.

        cout << dieFace << "       " << _____________________________ << endl;
    }
    return 0;
}
```

Listing 11: Code for Q1

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

Listing 12: Sample Output for Q1

Q2. (*4 marks*) Complete the code in listing 13 that asks the user for the number of array elements, then asks the user to enter values into that array. Finally, the program prints the histogram for the numbers in the array. Sample output is shown in listing 14. Assume that the values in the array `arr` are in the range from 0 to 4 only. The program uses *dynamic allocation* and *deletion* for array `arr`.

```
void printHistogram(int *arr, const int size){
    // 'freq' array stores the number of times each number in array 'arr' is repeated,
    int freq[5] = {0};

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

        _____;
    }
    // loop through all elements in array 'freq' to print histogram using 'ptr'.

    int *ptr = freq; // 'ptr' points to array 'freq'.

    for(int i = 0; i < 5; i++, ptr++) {
        cout << i << "            ";
        // Use 'ptr' variable only to access the 'freq' array to print stars
        // according to the frequency.

        for(int j = 0; _____ ; j++)
            cout << "*";
        cout << endl;
    }
}
void main() {
   int elements;
   cout << "Enter number of elements: ";
   cin >> elements;

   // Use 'new' keyword to dynamically allocate an array of integers of size 'elements'.

   int *arr = _____;

   cout << "Enter values for the array in the range (0-4): ";
   for(int i = 0; i < elements; i++)
       cin >> arr[i];

   // call function 'printHistogram'.
   printHistogram(arr, elements);

   // Use 'delete' keyword to dynamically delete the array 'arr'.

   _____;
}
```

Listing 13: Code for Q2

```
Enter number of elements: 15
Enter values for the array in the range (0-4): 1 2 3 4 0 1 1 4 2 4 2 2 2 2 1
0           *
1           ****
2           ******
3           *
4           ***
```

Listing 14: Sample Output for Q2

Q3. (*4 marks*) Complete the code in listing 15 that reads from the user 5 exam grades for 4 students, then computes the *minimum grade* for all 20 exam grades using `minimum` function and computes the *average grade* for each student using `average` function.

```
const int students = 4; // number of students.
const int exams = 5; // number of exams for each student.

void inputGrades(int res[][exams]){
    // Loop through all students and ask the user to enter their grades.
    for(int row = 0; row < students; row++) {
        cout << "Enter grades for student " << row << ": ";
        for (int col = 0; col < exams; col++) {
            // Use cin to ask the user to enter a grade 'col' for student 'row'.

            _____________________________________;
        }
    }
}
int minimum(int res[][exams]) {
    int min = 100; // initialize to highest possible grade
    for(int i = 0; i < students; i++)
        for(int j = 0; j < exams; j++) {
            // Check if the grade 'j' for student 'i' is less than the current 'min'.

            if(_____________________________________)

                min = res[i][j];
        }
    return min;
}
double average(int stGrades[]) {
    double total = 0;
    // total all grades for one student
    for(int i = 0; i < exams; i++)

        ___________________________________________________________;

    return (total/ exams);
}
void main() {
    // define an integer array 'results' for '4' students and '5' exams for each student.
    int results[students][exams];

    // ask user to enter grades for each student by calling 'inputGrades'
    inputGrades(results);

    // determine minimum grade by calling 'minimum' function.
    cout << "\nLowest grade: " << minimum(results) << endl;

    // calculate average grade for each student
    for (int i = 0; i < students; i++) {
        cout << "The average grade for student " << i << ": ";

        // Call function 'average' to compute the average grade for student 'i'.

        cout << _____________________________________ << endl;
    }
}
```

Listing 15: Code for Q3

Q4. (*3 marks*) Complete the code in listing 16 that sorts the array `arr` *descendingly* (from the largest to the smallest element) using *bubble sort* algorithm.

```
// swap values at memory locations to which 'ptr1' and 'ptr2' point
void swap(int *ptr1, int *ptr2)
{
    int temp = *ptr1;

    ____________________________________;

    *ptr2 = temp;
}

// sort an array of integers using bubble sort algorithm
void bubbleSort(int *array, const int size)
{
    // loop to control passes
    for (int k = 0; k < size - 1; k++)
    {
        // loop to control comparisons during each pass
        for (int i = 0; i < size - 1; i++)
        {
            // Check if element 'i' is less than element 'i + 1'.

            if(____________________________________)
            {
                // call function swap to swap the elements 'i' and 'i + 1'.

                swap(____________________________________);
            }
        }
    }
}

int main()
{
    const int arraySize = 10;
    int arr[arraySize] = {2, 6, 4, 8, 10, 12, 89, 68, 45, 37};

    cout << "Data items in original order: ";
    for(int j = 0; j < arraySize; j++)
        cout << " " << arr[j];

    bubbleSort(arr, arraySize);  // sort the array

    cout << "\nData items in descending order: ";
    for(int j = 0; j < arraySize; j++)
        cout << " " << arr[j];
    cout << endl;
    return 0;
}
```

Listing 16: Code for Q4

```
Data items in original order:  2 6 4 8 10 12 89 68 45 37
Data items in descending order:  89 68 45 37 12 10 8 6 4 2
```

Listing 17: Sample Output for Q4

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