CPE 150
INTRODUCTION TO PROGRAMMING
SECOND “MAKEUP” EXAM

Department of Computer Engineering
Yarmouk University
August 10, 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 4 questions, and 6 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: ________________________________

Q1. (*4 marks*) A string is *palindrome*, if it can be read from left to right the same as from right to left. For example, `madam` is a palindrome string, however, `osameh` is not. Complete the following recursive function `isPalindrome` to check whether the given `char` array `str` is a *palindrome* string.

```
Enter a word: madam
'madam' is a palindrome string.
```

Listing 1: Sample Output 1

```
Enter a word: qwerty
'qwerty' is not a palindrome string.
```

Listing 2: Sample Output 2

Answer for Q1:

```
int isPalindrome(char str[], int i, int j){


}

int main() {
    char str[200];
    cout << "Enter a word:  ";
    cin >> str;
    if(isPalindrome(str, __________, __________))
        cout << "\'" << str << "\'" << "is a palindrome string." << endl;
    else
        cout << "\'" << str << "\'" << "is not a palindrome string." << endl;
    return 0;
}
```

Q2. (*4 marks*) Complete the following C++ function `printUnique` to only print the *unique* values in the given array `arr`. A value in an array is *unique* if it is repeated one-time only in the array. The range of the values in the array `arr` is only between 0 and 20.

```
Enter 15 values for an array:
1 2 3 4 4 3 2 1 6 3 4 9 8 0 12
Unique values: 0 6 8 9 12
```

Listing 3: Sample Output 1

```
Enter 15 values for an array:
11 2 3 4 4 13 2 6 3 4 9 8 0 2 5
Unique values: 0 5 8 9 11 13
```

Listing 4: Sample Output 2

Answer for Q2:

```
void printUnique(int arr[], int size) {















}

int main() {
    int arr[15] = {0};
    cout << "Enter 15 values for an array:  " << endl;
    for(int i = 0; i < 15; i++)
        cin >> arr[i];
    printUnique(arr, 15);
    return 0;
}
```

Q3. (*4 marks*) Two arrays are called *anagrams* if both have the same values but with different order. In other words, two arrays have the same values and the frequency for each contained value is the same in both arrays. Complete the following C++ function `isAnagram` to check whether the arrays `arr1` and `arr2` are anagrams. Both arrays have the values in the range from 0 to 9.

```
Values for array1: 1 2 3 3 4 5 5 6 7 0
Values for array2: 0 7 5 6 5 3 4 3 1 2
Arrays are anagram!
```

Listing 5: Sample Output 1

```
Values for array1: 1 2 3 3 4 5 5 6 7 0
Values for array2: 0 0 0 6 5 3 4 3 1 2
Arrays are not anagram!
```

Listing 6: Sample Output 2

Answer for Q2:

```
int isAnagram(int arr1[], int arr2[], int size) {















}

int main() {
    int arr1[10], arr2[10];
    cout << "Enter values for array1:  ";
    for(int i = 0; i < 10; i++)
        cin >> arr1[i];
    cout << "Enter values for array2:  ";
    for(int i = 0; i < 10; i++)
        cin >> arr2[i];
    if(isAnagram(arr1, arr2, 10))
        cout << "Arrays are anagram!" << endl;
    else
        cout << "Arrays are not anagram!" << endl;
    return 0;
}
```

Q4. (*3 marks*) Complete the following function `shuffle` to shuffle the content of the array `arr`. The shuffling operation iterates through the array elements and at each element i it uses `rand` function to generate a random index j so that the element at index i is swapped with the element at index j.

```
Values for array: 1 2 3 3 4 5 5 6 7 0
Shuffled array: 0 7 6 5 4 3 5 1 2 3
```

Listing 7: Sample Output 1

```
Values for array: 0 0 0 6 5 3 4 3 1 2
Shuffled array: 0 1 2 3 5 6 3 4 0 0
```

Listing 8: Sample Output 2

Answer for Q4:

```
void shuffle(int arr[], int size) {






















}

int main() {
    int arr[10];
    cout << "Values for array:  ";
    for(int i = 0; i < 10; i++)
        cin >> arr[i];
    shuffle(arr, 10);
    cout << "Shuffled array:  ";
    for(int i = 0; i < 10; i++)
        cout << arr[i] << " ";
    cout << endl;
    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 / **Example**

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

if/else / **Example**

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

if/else if / **Example**

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

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

Form / **Example**

```
if (expression)              if (x < y)
{                            {
   statement;                   x++;
   statement;                   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: / **Example:**

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

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

*The do-****while*** *Loop*

Form: / **Example:**

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

do                       do
{                        {
   statement;               cout << x << endl;
   statement;               x++;
} while (expression); } 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