B.Sc. Engg. CSE 3rd Semester 08 March 2019 (Afternoon)

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
Department of Computer Science and Engineering (CSE)

MID SEMESTER EXAMINATION **WINTER SEMESTER, 2018-2019**

DURATION: 1 Hour 30 Minutes **FULL MARKS: 75**

CSE 4301: Object Oriented Programming

Programmable calculators are not allowed. Do not write anything on the question paper.

There are **4 (four)** questions. Answer any **3 (three)** of them.

Figures in the right margin indicate marks.

---

1. Create a class called **StudentInfo** where the details of a student's academic records are stored. In the above mentioned class, the private properties will include the name and ID of the student. Moreover, a student can take any number of courses and the results of the courses are to be kept in an integer array inside **StudentInfo** class. All these properties need to be private and can only be accessed outside of **StudentInfo** class by **CalculateAvgResult** method from **ResultService** class and through the parameterized constructor of the **StudentInfo** class. It should be noted that the properties of the **StudentInfo** class can only be set once through this its constructor as parameters and the values should be immutable. **CalculateAvgResult** method calculates the average result of a given student. Moreover, ResultService include another method called **SortStudents** which takes an array of students as parameter and sorts them according to their average results in ascending order. Finally, **SortStudents** prints the names of the students according to the sorted list. 25

   Note: The **interface** and **implementation** for both the classes should be in separate files with each file marked elaborately along with their names. Moreover, you should include a main function in a separate file to demonstrate your implementation. Each file should refer to all required header and library files according to necessity.

2. a) How Structured programming is different from Object Oriented Programming? 5

   b) Write short notes on 4x2

      i. Function Overloading
      ii. Encapsulation
      iii. Polymorphism
      iv. Inheritance

   c) What are the purposes of Destructors in C++ classes? With proper example describe the problems associated with Destructors while assigning objects, passing objects as parameters to a function and returning object from a function. Also provide worked out solution to such problems. 12

3. Check out the code snippet in Figure 1. Complete the classes according to the instruction given as comments. Given two character-arrays, use the first array to build your ***dictionary***. Each ***'element'*** object holds unique characters from the first array as ***key,*** and the number of times they appear in the first array is stored in ***count.*** The second array is used to decrease counts of corresponding ***element keys***. You cannot use any library other than ***iostream.*** 25

   For example, if you were given two character-arrays: **ABACB** and **BCCGG,** you should at first build your dictionary which should be like **(A: 2), (B: 2), (C: 1)**. In the second array B and C occurs once and twice respectively. You should decrease the count of B and C from your dictionary. The final status of the dictionary should be like this: **(A: 2), (B: 1).** Notice that as C appears twice, the ***key*** **'C'** is removes from dictionary. Also there was no ***key*** **'G'** in the initial ***dictionary*** and thus has no impact on the ***dictionary***.

   Demonstrate your code by writing a ***main function*** that takes two character-arrays of arbitrary length from user and prints the final status of the ***dictionary***.

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

class element {
    char key;
    int count;
public:
    /*Implement appropriate constructors, getters and setters*/
};

class dictionary {
    /*Declare an object pointer of 'element' class and other properties
as necessary. */
public:
    /*Create necessary constructors. */
    /*Check if element with the given key exists. */
    bool hasElement(char key);
    /*Add a new element to the array of 'element' objects. Return false
if element already exists. Else return true. */
    bool addElement(element elm);
    /*Add a new element to the array of 'element' objects. Return false
if element already exists. Else return true. */
    bool addElement(char key, int count);
    /*remove element from array of 'element' objects. Return false if no
such element exists. Else return true. */
    bool removeElement(element elm);
    /*remove element from array of 'element' objects by matching the
key. Return false if no such element exists. Else return true. */
    bool removeElement(char key);
    /*Increase count of the element that matches the key. Return false
if no such element exists. Else return true. */
    bool increaseCount(char key);
    /*Decrease count of the element that matches the key. Return false
if no such element exists. Else return true. If element's count
decreases to zero, remove that particular element. */
    bool decreaseCount(char key);
    /*Print all existing elements along with their count. */
    void printElements();
};
```

Figure 1

4. a) Check the code snippet given in Figure 2. Complete the code according to the comments provided. 10

```
class classA {
    int i;
    float f;
public:
    /*Write exactly THREE constructors. No other methods are allowed*/
};

int main() {
    int count = 4;
    /*In ONE executable C++ statement create and initialize an array of
classA objects with 'count' number of elements (in this case [count=4]).
    The first object should have {i=0, f=0.0}
    The second object should have {i=1, f=0.0}
    The third object should have {i=0, f=2.0}
    The fourth object should have {i=3, f=4.5}*/
    return 0;
}
```

Figure 2

b) Figure 3 is a complete program. What would be the output of the program? If you think there will be any possible anomalies in the output, mention them with proper explanations. 10

```
class test {
    int count;
    int *p;
public:
    test() {
        this->count = 5;
        this->p = new int[this->count]{ 0,0,0,0,0 };
    }
    test(int count, int* p) { this->set_p(count, p); }
    void set_p(int count, int* p) {
        this->count = count;
        this->p = new int[this->count];
        for (int i = 0; i < count; i++)
            this->p[i] = p[i];
    }
    int* get_p() {return this->p;}
    test* getTest() {return this;}
};
int set(test* obp) {
    int count = 5;
    int* p = new int[count] {10, 20, 30, 40, 50};
    obp->set_p(count, p);
    return count;
}
int set(test obp) {
    int count = 6;
    int* p = new int[count] {1, 2, 3, 4, 5, 6};
    obp.set_p(count, p);
    return count;
}
void print(test* p, int count) {
    int i = 0;
    while (true) {
        cout << *(p->get_p() + i) << ' ';
        i++;
        if (i >= count) break;
    }
    cout << endl;
}
int main() {
    int count = 4;
    int* p = new int[count] {100, 200, 300, 400};
    test* obP = new test(count, p);
    print(obP, count);
    count = set(obP);
    print(obP, count);
    count = set(*obP);
    print(obP, count);
    test ob = *obP->getTest();
    print(&ob, count);
    return 0;
}
```

Figure 3

c) Revisit the program in Figure 3 of Question 4(b). What impact would there be if the class ***test*** included a destructor with the following code in Figure 4? Would there be any run-time exception? If yes, then why? 5

```
~Test(){delete[] p;}
```

Figure 4