MULTIMEDIA UNIVERSITY

STUDENT ID NO

VENUE : _____

SEAT NO : _____

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 3, 2018/2019

TCP 1201 – OBJECT-ORIENTED PROGRAMMING AND DATA STRUCTURES

(All sections / Groups)

31 May 2019
9.00 a.m. – 11.00 a.m.
(2 Hours)

Question	Mark
1	
2	
3	
4	
Total	

INSTRUCTIONS TO STUDENT

1. This question paper consists of 15 printed pages (including cover page).

2. Attempt all questions. The distribution of the marks for each question is given.

3. Print all your answers **CLEARLY** in the specific answer box provided for each question. Submit this question paper at the end of the examination.

QUESTION 1 [10 marks]

(a) Explain the terms **rule of three** and **rule of five** in C++.

[2 Marks]

(b) The following main function calls the function SQRT with an int parameter x. When the user enters -1 the program returns [sqrt(-1) = **nan**]. Rewrite the program using C++ exceptions to prevent this from happening.
The SQRT function must throw a ***runtime_error*** with a message ("-ve number"). The catch block must capture the exception and display the error message.

[3 Marks]

```
#include <iostream>
#include <cmath>
using namespace std;

double SQRT(int x)
{
    double s = sqrt(x);
    return s;
}

int main()
{
  int x;
  cout<<"enter a value:";
  cin>>x;
  double y;
  y=SQRT(x);
  cout << "sqrt(" << x
       <<") = " << y
       << endl;

  return 0;
}
```

Answer:

Continued ...

(c) What is the output of the following program?

[2 Marks]

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

class A {
private:
    int a;
public:
    A(int a=0):a(a){}
    virtual void get(int a){
        this->a=a;
    }
    virtual void display(void){
        cout << "Value of a: "
             << a << endl;
    }
};
class B: public A {
private:
    int b;
public:
  B(int b=0):A(-1),b(b){}
  virtual void get(int a,int b)
  {
    A::get(a);
    this->b = b;
  }
  virtual void display(void) {
     A::display();
     cout << "Value of b: "
          << b << endl;
  }
};
```

```
class C: public B {
private:
    int c;
public:
C(int c=0):B(-2),c(c){}
void get(int a,int b,int c)
{
   B::get(a,b);
   this->c=c;
}
void display(void) {
   B::display();
   cout << "Value of c: "
        << c << endl;
}
};
int main(){
    C objC;
    objC.get(10,20,30);
    objC.display();

    A *ptr = new C;
    ptr->get(-5);
    ptr->display();

    return 0;
}
```

Answer:

Continued ...

(d) A Company employs employees (who can work only for one company). A company may have one or more departments. Each company has a dedicated president, who is an employee in the company. An employee can work only in one department. The departments run one or more projects. Employees can work in 1 to 3 projects, while a project can have 2 to 25 assigned employees. You may assume that a company has a name and address, while employees have an employee number and a salary. A project has an id, duration, and a list of employees assigned to it.

Draw the corresponding UML diagram showing the classes, their attributes, relations, and multiplicities.

[3 Marks]

Answer:

Continued ...

QUESTION 2 [10 marks]

(a) Explain with example the terms overloading and overriding in C++.

[2.5 Marks]

Answer:

Continued ...

(b) Write an implementation for the addition operator (+) as a friend function to the Complex class. The overloaded function should add **re** of the first object to **re** of the second object, and **im** of the first object to the **im** of the second object then returns the result as a Complex object.

Assume all the header files are included and the namespace std is used.

```
class Complex {
private:
    double re;
    double im;
public:
    Complex(float r,float i) {
        re = r;
        im = i;
    }
    Complex(float r) {
        re = r;im = 0.0;
    }
    double Magnitude(){
        return sqrt(re*re + Imag()*Imag());
    }
    double Real() { return re; }
    double Imag() { return im; }
};
```

[2 .5 Marks]

Answer:

Continued ...

(c) Write a complete C++ implementation for the move constructor of the Complex class below.

Assume all the needed include files are included and the std namespace is used.

```
class Complex {
private:
    double *re;
    double *im;
public:
    Complex(float r,float i) {
        re = new double;
        *re = r;
        im = new double;
        *im = i;
    }
    double Magnitude(){
        double mre = (*re)*(*re);
        double mimag = Imag()*Imag();
        return mre+mimag;
    }
    double Real(){return *re;  }
    double Imag(){ return *im; }
};
```

[2.5 Marks]

Answer:

Continued ...

(d) A compilation error occurs when compiling the program in the main function. Identify the error/errors and correct the lines that are involved. You are not allowed to alter any of the classes' definitions. Assume all the needed include files are included and the **std** namespace is used.

[2.5 Marks]

```
class Person {
    string name;
public:
 Person(string s="default")
 {
    name = s;
 }
 void setName(string nm) {
    name = nm;
 }
 string getName(){
    return name;
 }
 virtual void show() {
  cout << name  << endl;
 };
};
class Engineer:public Person
{
  string major;
public:
 Engineer(string nm,
          string maj){
    major = maj;
    setName(nm);
 }
 void show(){
    cout << getName() << ":"
         << major << endl;
 }
};
```

```
class Student:public Person{
    int id;
public:
 Student(string nm, int d){
   id = d;
   setName(nm);
}
void show(){
 cout << getName() << ":"
      << id << endl;
}
};
int main(){
 srand(unsigned(time(0)));
 Person *arr[] = {
   new Student("A",10),
   new Engineer("B","civil")
 };
 int index = rand()%2;

 Student *st = arr[index];
 st->show();

 return 0;
}
```

Answer:

Continued ...

QUESTION 3 [10 marks]

(a) Explain the major differences between the set, multiset, and the unordered_set classes of the STL with respect to the order of storing elements in the data structure and allowing duplicates.

[1.5 Marks]

	set class	multiset class	unordered_set
Order of element			
Allows duplicates			

(b) What is the output generated after running the following program. The program uses the STL stack and vector classes.

[1.5 Marks]

```
#include <iostream>
#include <stack>
#include <vector>

using namespace std;

int main() {
 vector<int> v1={1,2,3,4,5};
 stack<int> s1, s2;
 for (auto x:v1){
    if (x%2 == 0)
       s1.push(x);
    else
    s2.push(x);
 }
 vector<int> v2;
 while(!s1.empty()){
    v2.push_back(s1.top());
    s1.pop();
 }
 while(!s2.empty()){
    v2.push_back(s2.top());
    s2.pop();
 }
 for (auto x:v2){
    cout << x << " : ";
 }
 return 0;
}
```

Answer:

Continued ...

(c) Rewrite the following C++ program using templates so that the main function can compile and run without errors. You can skip the member functions that are not changed.

[2.5 Marks]

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

class MyArray{
    int size;
    int *data;
public:
  MyArray(int sz = 5){
   size = sz;
   data = new int(size);
  }
  int getSize(){
    return size;
  }
  int getData(int index){
    return data[index];
  }
  ~MyArray(){
     delete data;
  }
};
void print(MyArray& mr){
  cout << "Data = "
       << endl;
  for(int i=0;
      i<mr.getSize();i++)
    cout << mr.getData(i)
         << endl;
}
int main() {
    MyArray<int> t(5);
    print(t);
    MyArray<float> *ptr;
    ptr=new MyArray<float>(4);
    print(*ptr);
    delete ptr;
    return 0;
}
```

Answer:

Continued ...

(d) The following code is an implementation of the queue data structure using circular dynamic array. Implement the method **enqueue** which adds a new item to the queue. If the queue is full the function must display a message "Queue is full".

[2.5 Marks]

```
class MQueue {
private:
    int *queueArray;
    int queueSize;
    int front;
    int rear;
    int numItems;
public:
    MQueue(int qs);
    MQueue (const MQueue &qq);
    ~MQueue ();
    bool isEmpty()const;
    bool isFull() const;

    void enqueue(int v);

    int dequeue();
    int size();
    void clear();
};
MQueue::MQueue(int qs){
    queueSize = qs;
    queueArray = new int [qs];
    front = 0;
    rear = 0;
    numItems = 0;
}
```

Answer:

Continued ...

(e) Write a recursive implement for the **Counter** function below so that it returns the number of occurrences of the character **ch** in a string **str**.

[2 Marks]

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

int Counter(string str, char ch){

}
int main() {
    string str = "MMUFOEFCI";
    char c = 'F';
    cout << c << " occured "
         << Counter(str,c)
         << " times\n";
    return 0;
}
```

Output:

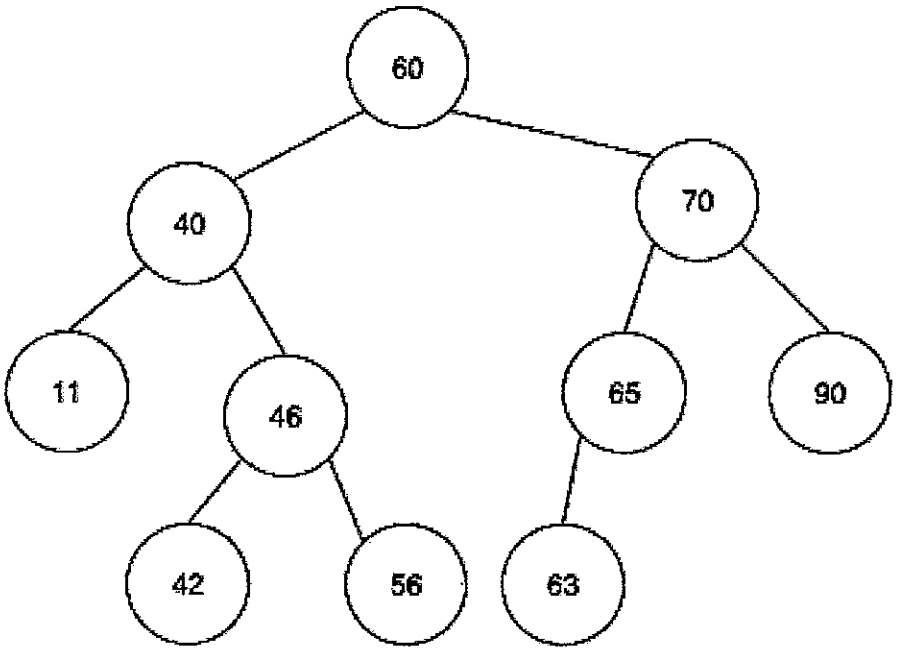
F occurred 3 times

Answer:

Continued ...

QUESTION 4 [10 marks]

(a) Consider the following **binary search tree**. Use the original tree to answer sub sections (1) to (5).



[3 Marks]

(1)	If the value 54 is inserted into this tree, which node becomes its parent?
(2)	If the value 27 is inserted into this tree, which node becomes its parent?
(3)	If we delete node 60, which node should be its replacement node?
(4)	If we delete node 46, which node should be its replacement node?
(5)	Traverse the BST using pre-order traversal.

Continued ...

Given the following class definitions, answer the sub sections (b) to (d).

```
struct Node {
    int info;
    Node *next;
};

class List{
    Node *head;
    Node *tail;
public:
    List();
    void push_front(int value);
};
```

(b) Implement the default constructor to create an empty list.

[1 Mark]

```
Answer:
```

(c) Implement the push_front() method that will add a new node to the front of a list and increment the size by one.

[2.5 Marks]

```
Answer:
```

Continued ...

(d) Draw a UML class diagram that shows the classes, relationships and multiplicities exists in the C++ code given earlier (Node and List classes).

[1.5 Mark]

Answer:

(e) Discuss two uses of the **final** keyword in C++. Give an example line of code on how to use it.

[2 Marks]

Answer:

End of Page