MULTIMEDIA UNIVERSITY

STUDENT ID NO

					4	5	1	1	8

VENUE : ____________________

SEAT NO : ____________________

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 1, 2019/2020

TCP 1201 – OBJECT-ORIENTED PROGRAMMING AND DATA STRUCTURES

(All sections / Groups)

19th OCTOBER 2019
2.30 p.m. – 4.30 p.m.
(2 Hours)

Question	Mark
1	
2	
3	
4	
Total	

INSTRUCTIONS TO STUDENT

1. This question paper consists of 17 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) For object-oriented software, the conceptual framework is the object model. List four major characteristics of object-oriented programming.

[2]

b) Draw a UML class diagram for the software described below. You need to use appropriate name for the classes and label the relationship between classes with the correct annotations. You need to use OOP in the design whenever possible.

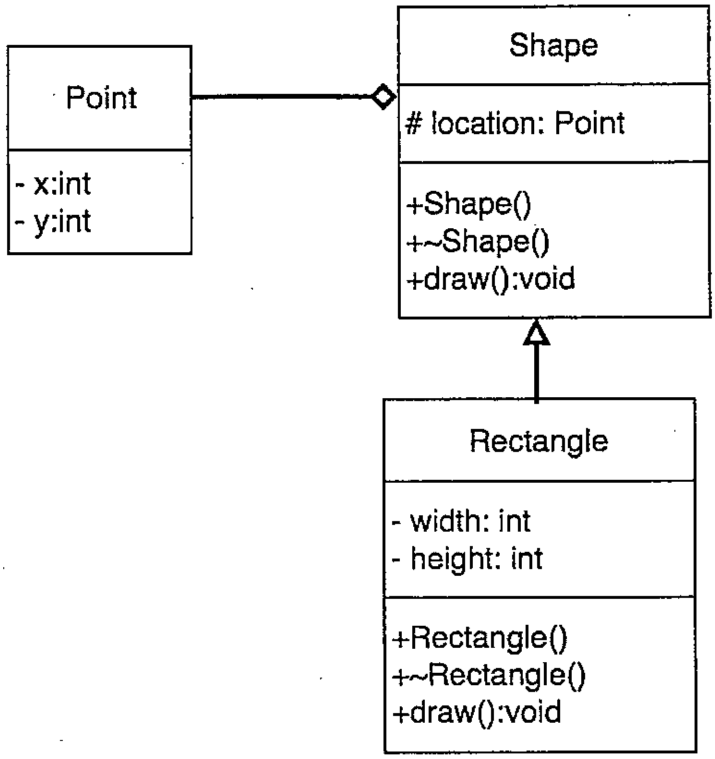
The software is a system to be used by MMU to manage subject registration of their students. There are several faculties in MMU. Each faculty contains two major components: the persons using the system, and the undergraduate programmes offered to the students. The persons can be either staffs or students with a valid address.

The undergraduate programmes has a name that is registered with the Ministry of Education. There are a list of subjects under the programme with a unique name and title. The subjects might have a pre-requisite requirement for enrollment.

[4]

c) Considering the UML class diagram and the main function below, provide the necessary C++ code for Shape and Rectangle classes to generate the output as shown.

[4]



```
int main() {
   Shape *pShape[] = {new Shape, new Rectangle };

   for(auto x:pShape)
      x->draw();
   for(auto x:pShape)
      delete x;

   return 0;
}
```

Output:

```
Shape is created
Shape is created
Rectangle is created
Shape is drawn
Rectangle is drawn
Shape is destroyed
Rectangle is destroyed
Shape is destroyed
```

Continued...

QUESTION 2: [10 marks]

a) Rewrite the `DayToWeek` class in order to get the `main` function to compile and generate the output shown below.

[4]

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

class DayToWeek{
private:
    int day;
    double week;
    void getWeek() { week = (double)day/7; }

public:
    DayToWeek() {
      day = 0;
      getWeek();
    }
    DayToWeek( int d) {
        day = d;
        getWeek();
    }
    void setDay( int d ) {
        day = d;
    }

    void print(){
        cout << "Day = "<< day << ", week =  "
        << week << endl;
    }
};


int main() {
    DayToWeek <double>d1(6.5);
    DayToWeek <int>d2(8);
    d1.print();
    d2.print();

    return 0;
}
```

```
Output:
Day = 6.5, week = 0.9
Day = 8, week = 1.2
```

Continued...

b) Given the `main` function below, provide the necessary C++ code in `DayToWeek` class to get the output as shown.

[3]

```
int main() {
    DayToWeek <float>d1(6.5);
    DayToWeek <float>d2(8.5);
    d1.print();
    d2.print();
    d1 = d1 + d2;
    d1.print();
    return 0;
}
```

```
Output:
Day = 6.50, week =  0.93
Day = 8.50, week =  1.21
Day = 15.00, week =  2.14
```

c) In real life problem, the `day` parameter must always be a positive value. Rewrite the `setDay` function to implement the exception handling for the negative `day` value.

[3]

QUESTION 3: [10 marks]

a) Complete the following recursive function which counts the number of occurrences of a `search` character in the `str` string variable.

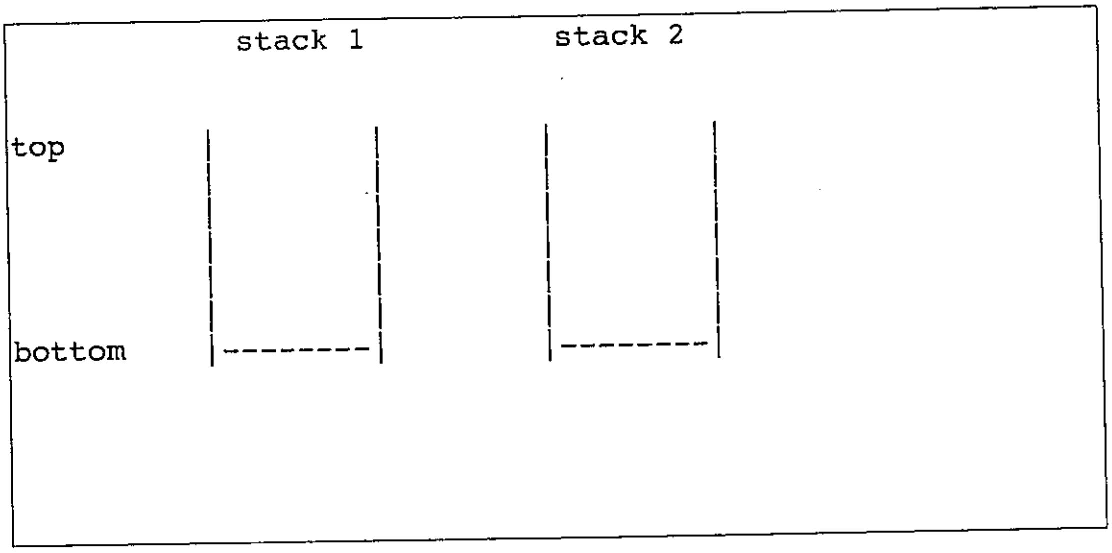
[4]

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

int fun( char search, char str[], int subscript)
{
  if( str[subscript] == '\0' )

    ______________________________
  else
  {

    if( ____________________________________ )
    {

      ____________________________________________
    else
    {

      ____________________________________________
    }
  }
}


int main()
{
  char str[] = "Cyberjaya";
  cout << fun('a', str, 0) << endl; // Output is 2

  return 0;
}
```

b) What are the contents of the initially empty stacks stack1 and stack2 after the following sequence of operations?

[2]

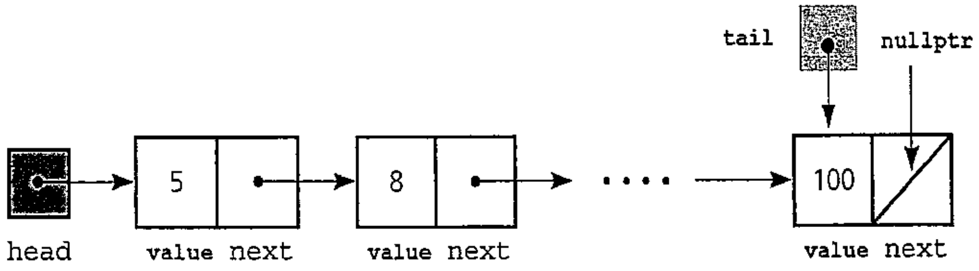
```
stack1.push(2)
stack1.push(3)
stack2.push(4)
stack2.push(5)
stack1.pop()
n = stack2.top()
stack1.push(n)
stack1.push(6)
stack2.push(7)
```



c) A pointer based linked list is made up of a sequence of nodes as shown in the following code and diagram. Based on the existing linked list given, describe in your own words and with the use of diagrams, how the `insert()` member function works when a new node is to be inserted in between the node with the value 5 and the node with the value 8. The use of C++ code or pseudo code is optional.

[4]

```
template<class T>
struct Node {
    T value;
    shared_ptr<Node<T>> next;
public:
    Node(int v){
        value = v;
        next = nullptr;
        cout << "Node is created...\n";
    }
    ~Node(){
        cout << "Node is destroyed ...\n";
    }
};

template<class T>
class MyList {
private:
    shared_ptr<Node<T>> head;
    weak_ptr<Node<T>> tail;

    int sz;

public:
    ...
    bool insert(int,T);
    ...
};
```



Continued...

QUESTION 4: [10 marks]

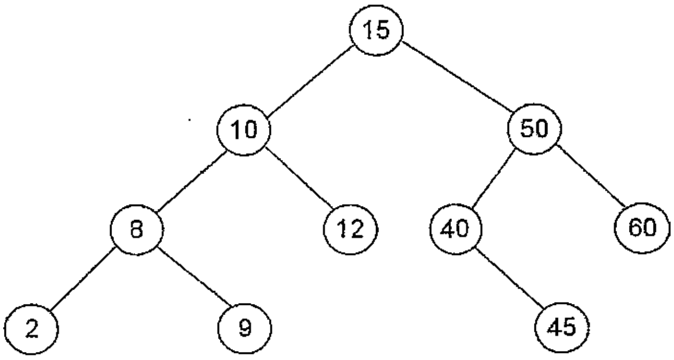
a) Given the class MQueue below, write the code to implement the enqueue( ) function of an array based queue outside the class body (not inline). [3]

```
template<class T>
class MQueue {
private:
    T *queueArray;
    int queueSize;
    int front;
    int rear;
    int numItems;
public:
    ...
    bool isFull() const;
    void enqueue(int v);
    ...
    ...
    ...
};

template<class T>
bool MQueue<T>::isFull() const{
    if (numItems >= queueSize)
        return true;
    else
        return false;
}
```

```
template<class T>
void MQueue<T>::enqueue(int v)
{












}
```

Continued...

b) Using the binary search tree shown below as the initial tree, show the binary search tree after:



(i) Insertion of 7, 1, 55 and 29 to the initial tree.

(ii) Deletion of 9, 40 and 10 from the initial tree. (Use inorder successors for these operations)

[4]

(i)

(ii)

c) Complete the following program which accepts ten integers and only sums up unique ones. The program must use an STL container. Choose an appropriate container for the program.

[3]

```
Sample Run 1
Integer 1 : 1
Integer 2 : 1
Integer 3 : 1
Integer 4 : 1
Integer 5 : 1
Integer 6 : 1
Integer 7 : 1
Integer 8 : 1
Integer 9 : 1
Integer 10 : 1
The sum of unique integers is 1.

Sample Run 2
Integer 1 : 1
Integer 2 : 1
Integer 3 : 1
Integer 4 : 1
Integer 5 : 2
Integer 6 : 2
Integer 7 : 3
Integer 8 : 4
Integer 9 : 4
Integer 10 : 4
The sum of unique integers is 10.
```

```
#include <iostream>

#include <_____________> // Include an STL container li-
brary
using namespace std;

int main()
{
   int input;

   __________________  // Declaration of an STL container

   for (int i = 0; i< 10; i++)
   {
      cout << "Integer " << i + 1 << " : ";
      cin >> input;

      //Store input into the STL container

      _____________________________________
   }
   // Write the code to calculate the sum of unique input
values.









   cout << "The sum of unique integers is "
        << sum << ". " << endl;
}
```