Second Semester Examination
2020/2021 Academic Session

July/August 2021

CPT113 – Programming Methodology & Data Structure

(Methodologi Pengaturcaraan & Struktur Data*)*

Duration : 2 hours
(Masa : 2 jam)

Please ensure that this examination paper consists of TWENTY ONE (21) pages of printed material before you begin the examination.

Instructions : Answer **ALL** questions in **SECTION A, B and C.** Section A and B are penalty-based marking. You will be given the stated mark for every correct answer and will be deducted half of the allocated mark for every wrong answer.

SECTION A : There are 10 questions in this section. Each question is 1 mark.

1. Given the partial class declaration:

```
class Author{
     private:
          string name, hometown;
          string *genreList;
     public:
          ...
```

Assume all variables are properly declared.
Demonstrate which of the following is the correct way of writing the constructor.

A.

```
Author(){
          name=""; hometown="";
          genreList = new string [GENRE_COUNT];
          for (int i=0; i<GENRE_COUNT; i++)
               genreList[i]="";
     }
```

B.

```
Author(string n, string h, string g){
          name=n; hometown=h;
          for (int i=0; i<GENRE_COUNT; i++)
               genreList[i]=g;
          }
```

C.

```
Author(){
          name=n; hometown=h;
          for (int i=0; i<GENRE_COUNT; i++)
               genreList[i]=””;
          }
```

D.

```
Author(string n, string h, string g){
          name=n; hometown=h;
          genreList = g
```

2. If a base class has a public member function, and the derived class has a member function with the same name with a different parameter list. Classify this function as __________.

A. syntax error

B. overloaded

C. overwritten

D redefined

3. Classify which of the following belongs to the practical application of the stack data type.

I. tracking nested loops in programming

II. storage of local variables in computer system

III. tracking nested function calls in computer system

IV. taking turn buying groceries in Tesco during COVID

V. completing task to do everyday

A. I and V

B. I, II and III

C. II, and V

D. III, IV and V

4. Demonstrate the number of times the following function call itself if **5** is passed as the argument?

```
void showMessage(int n){
     if (n > 0)
     {
          cout << "Good day!" << endl;
          showMessage(n + 1);
     }
}
```

A. Four

B. Five

C. Once

D. Infinite

5. When working with a binary tree, a node that has more than two children

_________.

A. is known as a triplet node

B. will be cut back by the compiler

C. is theoretically impossible in a correctly developed binary tree structure

D. None of these

6. Select the statement which is TRUE regarding object-oriented programming.

A. You must declare all data members of a class before you declare member functions

B. Class objects can be defined prior to the class declaration

C. Object encapsulates both the data and the functions that operate on the data

D. A public member function is useful for tasks that are internal to the class, but it is not directly called by statements outside the class

7. For the following code, select the statement that is **not** TRUE.

```
class Point
{
    private:
        double y;
        double z;
    public:
        double x;
};
```

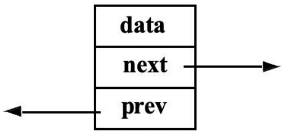
A. The name of the class is `Point`.

B. `x, y,` and `z` are called members of the class.

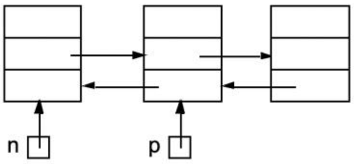
C. `x` is accessible to code that is written outside the class.

D. `z` is accessible to code that is written outside the class.

8. Nodes for a doubly linked list are defined to have the following structure:



The `next` instance variable stores a reference to the next node in the list, and the `prev` instance variable refers to the previous node in the list. Below is a list of three of these nodes, along with two reference variables, `n` and `p`, that refer to specific nodes in the list.



Select the expression that does not refer to the third node in the doubly linked list.

A. `n->next->next`

B. `p->prev->next`

C. `n->next->next->prev->next`

D. `p->prev->next->next`

9. Select which of the following statement is **not** TRUE about a doubly linked list.

A. We can navigate in both the directions

B. It requires more space than a singly linked list

C. The insertion and deletion of a node take a bit longer

D. Traversing in forward or backward manner is easier in a doubly linked list than a singly linked list

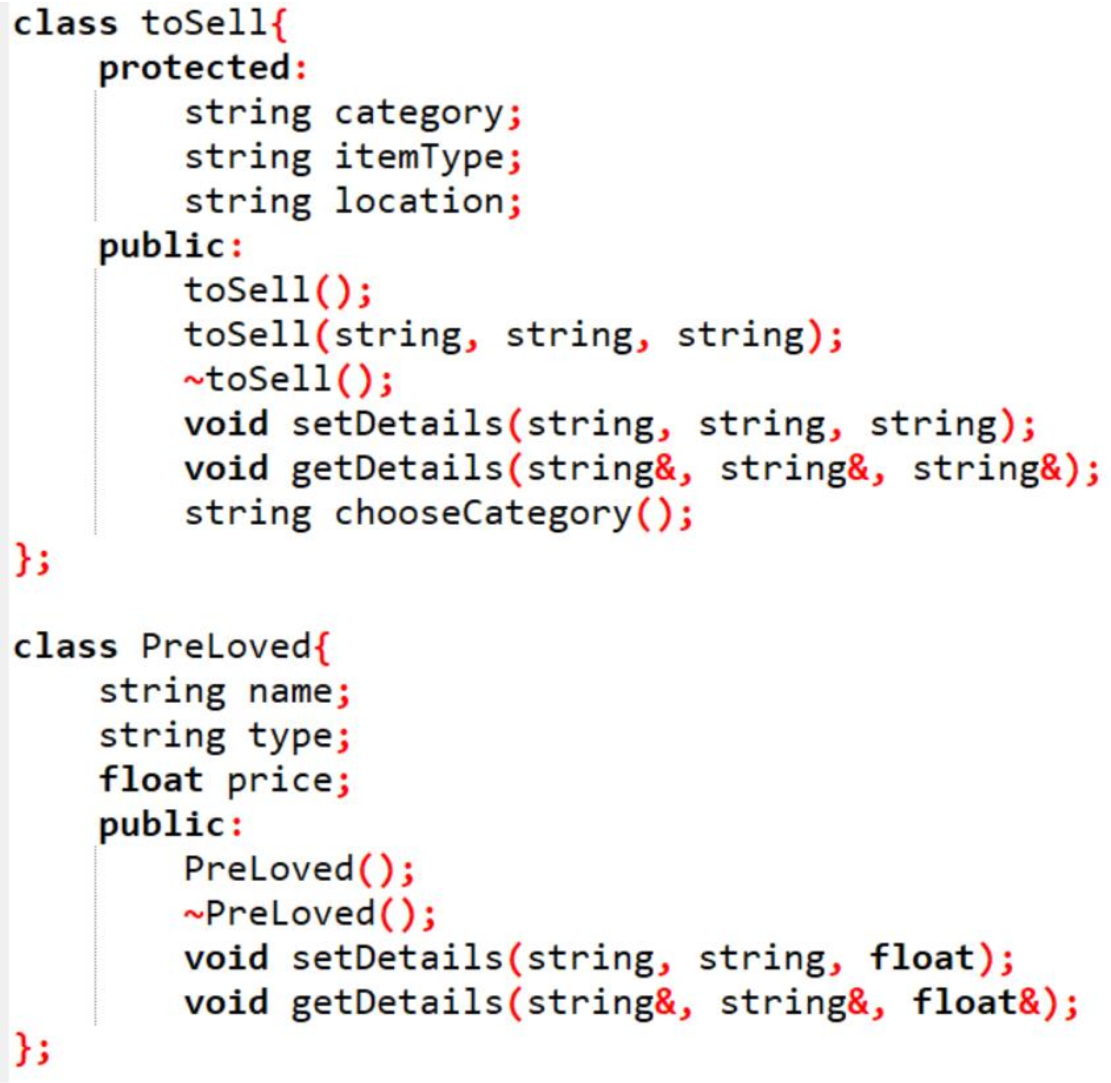
10. Given the following code, assume the `myQueue` object is a queue that can hold integers and that `value` is an `int` variable.

```
1   myQueue.enqueue(10);
2   myQueue.enqueue(20);
3   myQueue.enqueue(30);
4   myQueue.dequeue(value);
5   myQueue.dequeue(value);
6   myQueue.enqueue(value);
```

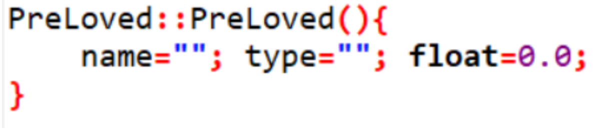
	`7    cout << value << endl;` Assume that the `dequeue` function, called on lines 4 and 5, stores the number removed from the queue in the `value` variable. Report what the statement on line 7 display.	
	A.	30
	B.	10
	C.	20
	D.	None of the above

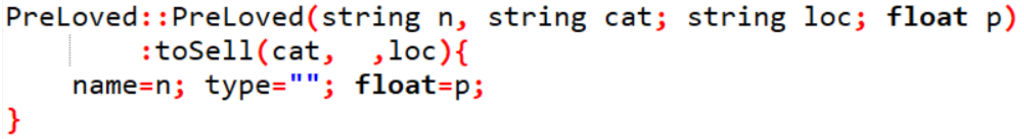
SECTION B : There are 10 questions in this section. Each question is 2 marks.

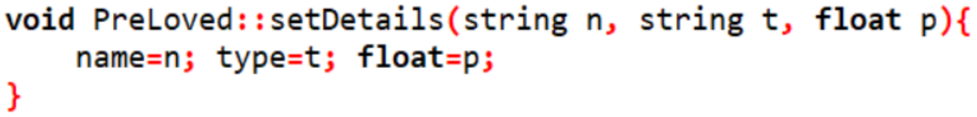
1. Given the following class declaration, show the correct overloading constructor for object PreLoved.

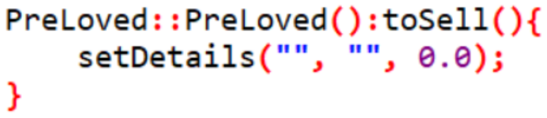


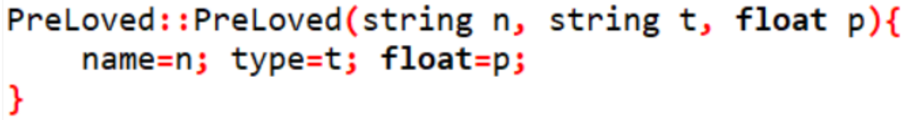
```
class toSell{
    protected:
        string category;
        string itemType;
        string location;
    public:
        toSell();
        toSell(string, string, string);
        ~toSell();
        void setDetails(string, string, string);
        void getDetails(string&, string&, string&);
        string chooseCategory();
};

class PreLoved{
    string name;
    string type;
    float price;
    public:
        PreLoved();
        ~PreLoved();
        void setDetails(string, string, float);
        void getDetails(string&, string&, float&);
};
```

	I.	
		`PreLoved::PreLoved(){`
		`    name=""; type=""; float=0.0;`
		`}`

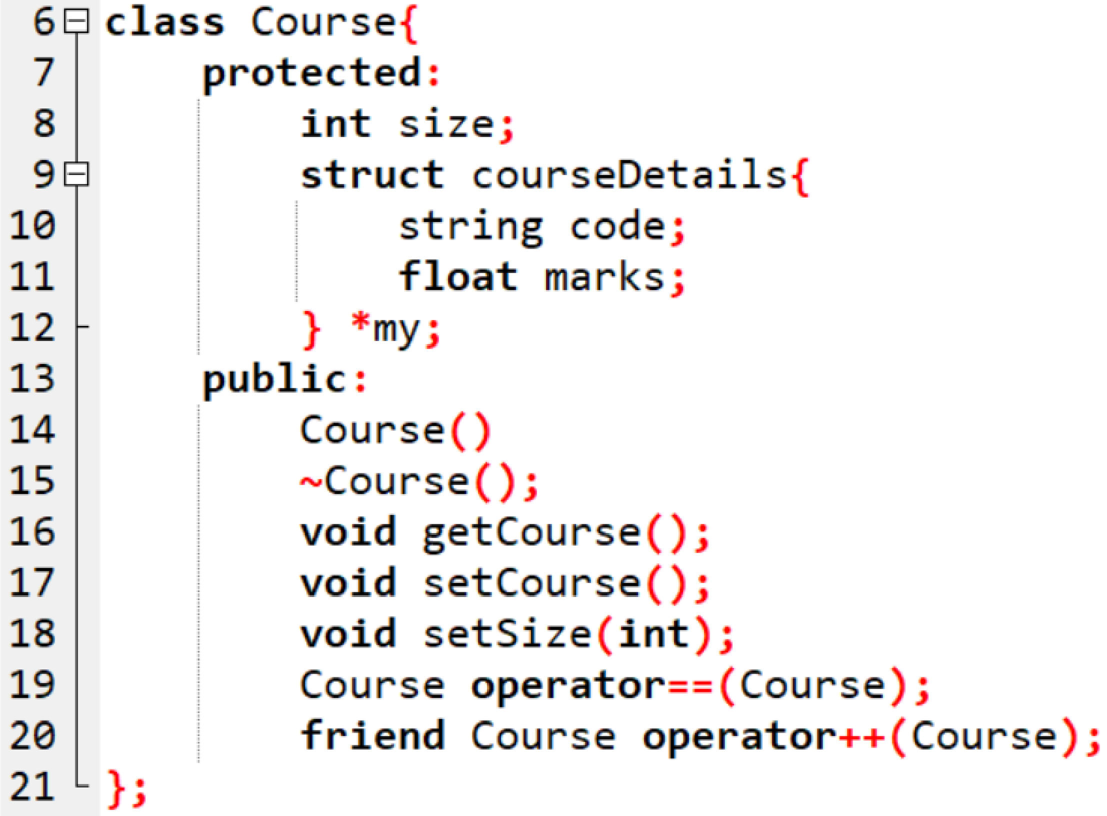
	II.	
		`PreLoved::PreLoved(string n, string cat; string loc; float p)`
		`        :toSell(cat,  ,loc){`
		`    name=n; type=""; float=p;`
		`}`

	III.	
		`void PreLoved::setDetails(string n, string t, float p){`
		`    name=n; type=t; float=p;`
		`}`

	IV.	
		`PreLoved::PreLoved():toSell(){`
		`    setDetails("", "", 0.0);`
		`}`

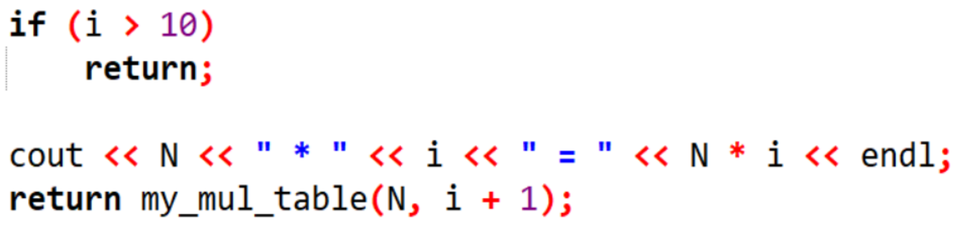
	V.	
		`PreLoved::PreLoved(string n, string t, float p){`
		`    name=n; type=t; float=p;`
		`}`

	A.	I, II, III and IV
	B.	I, IV and V
	C.	II, and IV
	D.	II and V
2.	Show member(s) of the following class declaration:	



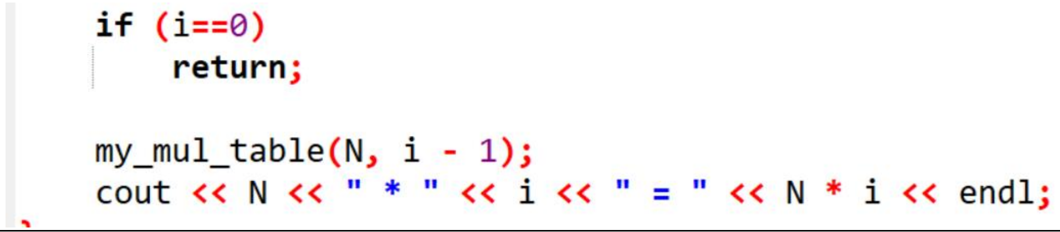
```
class Course{
    protected:
        int size;
        struct courseDetails{
            string code;
            float marks;
        } *my;
    public:
        Course()
        ~Course();
        void getCourse();
        void setCourse();
        void setSize(int);
        Course operator==(Course);
        friend Course operator++(Course);
};
```

	I.	`string code;`
	II.	`courseDetails *my;`
	III.	`~Course;`
	IV.	`course operator==(Course);`
	V.	`friend Course operator++(Course);`
	A.	I, II and IV
	B.	I and II
	C.	II, III, IV and V
	D.	II, III, and IV
3.	Given `N` is the number of a multiplication table and `i` is the index, which of the following recursive function is applicable to write incremental multiplication table?	
	I.	Given `i` is 1



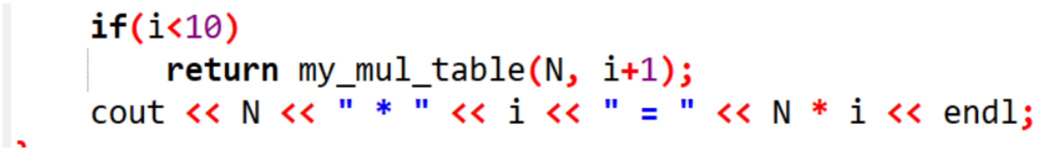
```
if (i > 10)
    return;

cout << N << " * " << i << " = " << N * i << endl;
return my_mul_table(N, i + 1);
```

II. Given i is 10



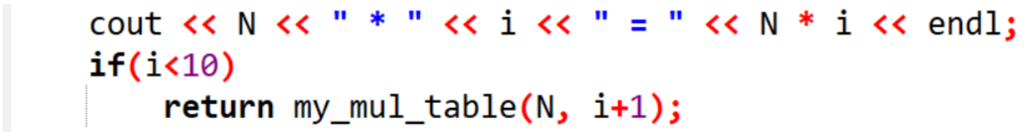
```
if (i==0)
    return;

my_mul_table(N, i - 1);
cout << N << " * " << i << " = " << N * i << endl;
```

III. Given i is 10



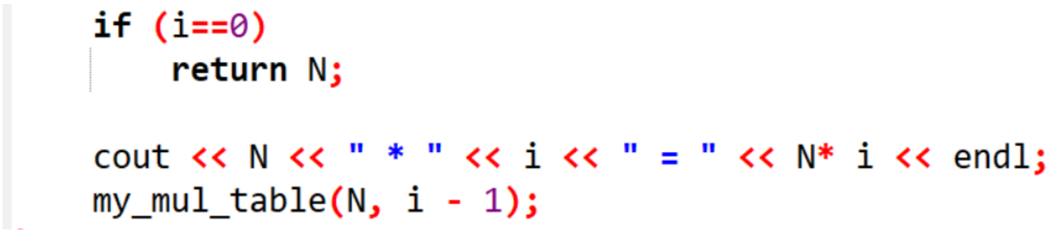
```
if(i<10)
    return my_mul_table(N, i+1);
cout << N << " * " << i << " = " << N * i << endl;
```

IV. Given i is 1



```
cout << N << " * " << i << " = " << N * i << endl;
if(i<10)
    return my_mul_table(N, i+1);
```

V. Given i is 10



```
if (i==0)
    return N;

cout << N << " * " << i << " = " << N* i << endl;
my_mul_table(N, i - 1);
```

A. I, II and III

B. I, II and IV

C. IV and V

D. II, III and V

4. Given an input sequence 1, 2, 3, 4, 5. Assuming this stack operate push and pop randomly. Illustrate all the possible incorrect output sequence in order to empty a stack.

	I.	3, 4, 5, 1, 2
	II.	3, 4, 5, 2, 1
	III.	1, 5, 2, 3, 4
	IV.	5, 4, 3, 1, 2
	A.	I and II
	B.	II and III
	C.	III and IV
	D.	I, III and IV
5.	Suppose that we have numbers between 1 and 100 in a **binary search tree** and want to search for the number 54. Illustrate which of the following sequences CAN be the sequence of nodes examined.	
	I.	{10, 75, 64, 43, 60, 57, 54}
	II.	{90, 12, 68, 34, 62, 45, 54}
	III.	{9, 85, 47, 68, 43, 57, 54}
	IV.	{79, 14, 72, 56, 16, 53, 54}
	A.	I and II
	B.	I, II and III
	C.	I, II, and IIV
	D.	All of the above
6.	Compute the following C++ codes, which correctly output the value 45.	

A.

```
#include <iostream>
using namespace std;
class TestClass
{
public:
     TestClass(int x)
     { cout << x << endl; }
     TestClass()
     { cout << "Hello!" << endl; }
};
int main()
{
     TestClass test(“45”);
     return 0;
}
```

B.

```
#include <iostream>
using namespace std;
class TestClass
{
public:
     TestClass(int x)
     { cout << x << endl; }
     TestClass()
     { cout << "Hello!" << endl; }
};
int main()
{
     TestClass test(45);
     return 0;
}
```

C.

```
  #include <iostream>
  using namespace std;
  class TestClass
  {
  private:
       int val;
       void showVal()
       { cout << val << endl; }
  public:
       TestClass(int x)
       { val = x; }
  };
  int main()
  {
       TestClass test(77);
       test.showVal();
       return 0;
  }
```

D.

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

```
{
public:
     TestClass(int x)
     { cout << “Hello” << endl; }
     TestClass()
     { cout << "Hello!" << endl; }
};
int main()
{
     TestClass test(45);
     return 0;
}
```

7. Demonstrate the valid constructor definitions for the following C++ codes.

```
class Employee
{
private:
   string name;         // Employee's name
   int idNumber;        // ID number
   string department;   // Department name
   string position;     // Employee's position

public:
   // Constructors
   Employee(string, int, string, string);
   Employee(string, int);
   Employee();

   // Mutators
   // Accessors

};
```

A.
```
Employee::Employee(string n, string i, string d,
string p)
   {
      name = n;
      idNumber = i;
      department = d;
      position = p;
   }
```

B.
```
Employee::Employee(string n, int i)
   {
      name = n;
      idNumber = i;
```

```
        department = "";
        position = "";
    }
```

C.

```
Employee::Employee()
    {
        name = "";
        idNumber = "";
        department = "";
        position = "";
    }
```

D. All of the above

8. Given the `IntList` class, demonstrate which one of the following that correctly insert a value `x` at position `y` in a linked list?

```
class IntList
{
private:
   struct ListNode
   {
      int value;
      struct ListNode *next;
   };
   ListNode *head;
   void destroy();

public:
   IntList()
      { head = nullptr; }
   IntList(const IntList &);
   ~IntList();
   void insert(int, int);
};
```

A.

```
void IntList::insert(int x, int y)
{
  ListNode *newNode = new ListNode;
  newNode->value = x;
  newNode->next = nullptr;

   if (head == nullptr)
   {
      head = newNode;
      return;
   }
   if (y == 0)
   {
      newNode->next = newNode;
      head = newNode->next;
     return;
   }

    ListNode *p = head;
    int num = 1;
    while (num <= y)
    {
       if (p->next == nullptr || num == y)
       {
         ListNode *tempPtr = p->next;
         p->next = newNode;
           newNode->next = tempPtr;
          return;
       }
      p = p->next;
      num++;
    }
```

B.

```
void IntList::insert(int x, int y)
{
  ListNode *newNode = new ListNode;
  newNode->value = x;
  newNode->next = nullptr;

   if (head == nullptr)
      {
           head = newNode;
           return;
      }
      if (y == 0)
    {
           newNode->next = head;
           head = newNode;
        return;
    }

    ListNode *p = head;
    int num = 1;
    while (num <= y)
    {
      if (p->next == nullptr)
       {
             ListNode *tempPtr;
              p->next = newNode;
              newNode->next = tempPtr;
          return;
       }
       p = p->next;
       num++;
    }
```

C.

```
void IntList::insert(int x, int y)
{
  ListNode *newNode = new ListNode;
  newNode->value = x;
  newNode->next = nullptr;

   if (head == nullptr)

      {
           head = newNode;
           return;
      }
      if (y == 0)
    {
           newNode->next = head;
           head = newNode;
        return;
    }

    ListNode *p = head;
    int num = 1;
    while (num <= y)
    {
       if (p->next == nullptr || num == y)
       {
            ListNode *tempPtr = p->next;
             p->next = newNode;
             newNode->next = tempPtr;
          return;
       }
       p = p->next;
       num++;
    }
```

D.

```
void IntList::insert(int x, int y)
{
  ListNode *newNode = new ListNode;
  newNode->value = x;
  newNode->next = nullptr;

   if (head == nullptr)

      {
           head = newNode;
           return;
      }
      if (y == 0)
    {
           newNode->next = head;
           head = newNode;
        return;
    }

    ListNode *p = head;
    int num = 1;
    while (num <= y)
    {
       if (p->next == nullptr || num == y)
       {
            ListNode *tempPtr = newNode;
            p->next = newNode;
            newNode->next = tempPtr;
            return;
       }
       p = p->next;
       num++;
    }
```

9. Given the following C++ codes segment on insertion a node into an ordered doubly linked list. Demonstrate the correct insertion case.

```
newnode = new ListNode<T>;
newnode->value = newItem;
newnode->next = nullptr;
newnode->previous = nullptr;

found = false;
nodePtr = head;

while(nodePtr != nullptr && !found){
   if (nodePtr->value >= newItem)
      found = true;
   else {
      trailPtr = nodePtr;
      nodePtr = nodePtr->next;
   }
}

if (nodePtr != nullptr)
{
  trailPtr->next = newnode;
  newnode->previous = trailPtr;
  newnode->next – nodePtr;
  nodePtr->previous = newnode;
}
```

A. Case 1: Insertion in an empty list

B. Case 2: Insertion at the beginning of a nonempty list

	C.	Case 3: Insertion at the end of a nonempty list
	D.	Case 4: Insertion somewhere in a nonempty list

10. Show the purpose of the following C++ codes.

```
void LinkedList<T>::processNode(T searchValue)
{
   ListNode<T> *nodePtr;
   ListNode<T> *previousNode;
   bool found = false;

   if (!head)
         cout <<"List is Empty\n";
   if (head->value == searchValue){
     nodePtr=head;
     head=head->next;
     head->next = nodePtr->next;
     delete nodePtr;
     count--;
   }
   else
   {
     nodePtr = head;
     while(nodePtr->value != searchValue && nodePtr->next !=
     head) {
         previousNode = nodePtr;
         nodePtr = nodePtr->next;
      }
     if (nodePtr->value == searchValue) {
        previousNode->next = nodePtr->next;
        delete nodePtr;
        count--;
     }
     else
        cout<<"Cannot delete the value "<<searchValue<<endl;
  }
}
```

	A.	Search and delete node from a doubly linked list
	B.	Search and delete node from a linked list
	C.	Search and delete node from a circular linked list
	D.	None of the above

SECTION C : Answer **ALL** questions.

1. Investigate the following problem:

A housing developer company builds 100 affordable houses. Each house consists of 4 rooms: a kitchen, two bedrooms and a living room. The cost of a house is based on the total area of rooms in the house. The company wants to calculate the total cost for all the houses.

Given the class `Rectangle` declaration:

```
class Rectangle
{
   private:
      double width;
      double length;
   public:
      void setWidth(double);
      void setLength(double);
      double getWidth() const;
      double getLength() const;
      double getArea() const;
};
```

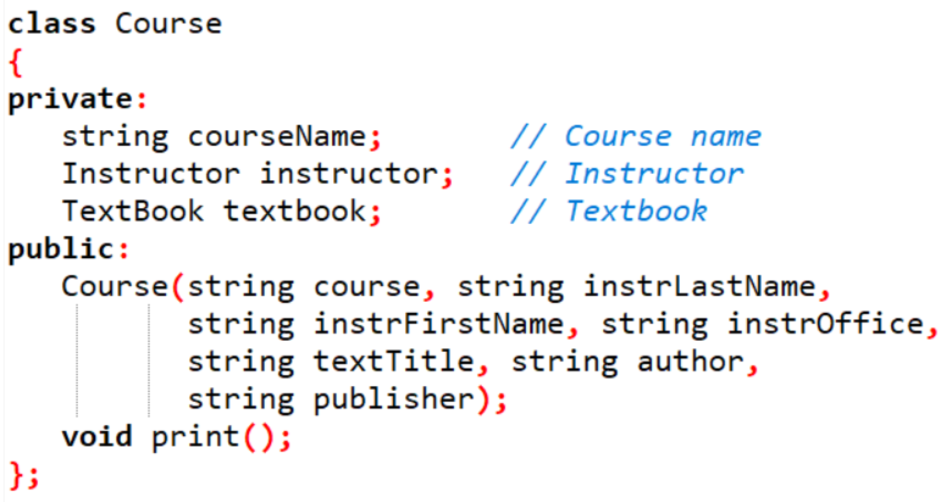
(a). Declare all the required object(s).

(5 marks)

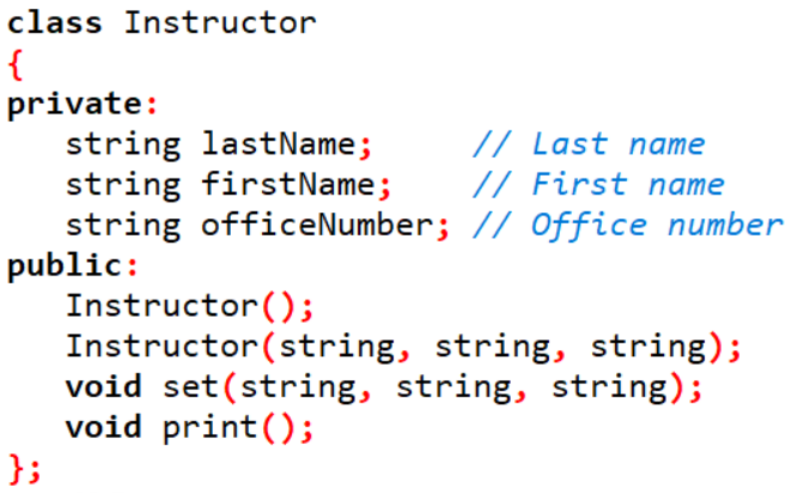
(b). Write the main C++ program to calculate the total cost of all the houses.

(10 marks)

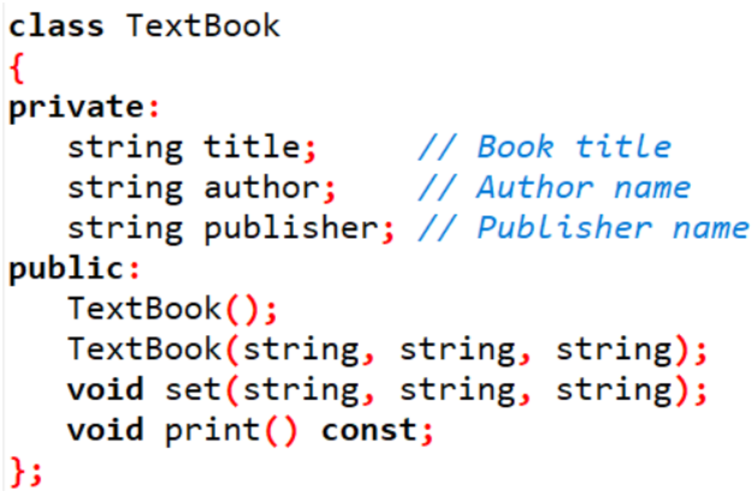
2. Examine the following class header:



```
class Course
{
private:
   string courseName;        // Course name
   Instructor instructor;    // Instructor
   TextBook textbook;        // Textbook
public:
   Course(string course, string instrLastName,
         string instrFirstName, string instrOffice,
         string textTitle, string author,
         string publisher);
   void print();
};
```



```
class Instructor
{
private:
    string lastName;      // Last name
    string firstName;     // First name
    string officeNumber; // Office number
public:
    Instructor();
    Instructor(string, string, string);
    void set(string, string, string);
    void print();
};
```



```
class TextBook
{
private:
    string title;      // Book title
    string author;     // Author name
    string publisher; // Publisher name
public:
    TextBook();
    TextBook(string, string, string);
    void set(string, string, string);
    void print() const;
};
```

	(a).	Based on the given function declaration given, write the complete constructor for class Course (4 marks)
	(b).	Write the function prints for all classes (7 marks)
	(c)	Write how to call the print function in main (2 marks)
3.		Analyse the following simplified COVID-19 vaccination system that has the following procedure and phases. **Step 1: Register** People fill up their details into the system which include name, IC, age, occupation, whether they have chronic disease and whether they have OKU status.

Step 2: Get Appointment Scheduled
The system process their details, determine their priority group and categorise their phase for the vaccination appointment date.

Phase A
Take place from February to April 2021, consisting of front liners:

- **Priority Group 1:** Front liners comprising of public and private healthcare personnel;
- **Priority Group 2:** Front liners consisting of essential services, defence and security personnel.

Phase B
Take place from May to August 2021, comprising people in high-risk groups:

- **Priority group 3:** Senior citizens aged 60 and over, those with chronic diseases, and OKU individuals.

Phase C
The final phase occur from September 2021 to February 2022, for the remainder:

- **Priority group 4:** Adult population aged 18 years and above.

(a) Distinguish the suitable data structures with justification.

(4 marks)

(b) Illustrate the classes by presenting the UML diagram.

(6 marks)

(c) Construct the complete C++ codes using OOP paradigm.

(15 marks)

4. Investigate the following program

```
bool myCode(string str, int a, int b){
     bool isTrue=false;
     do{
          if (str[a]== str[b]){
               a++;
               b--;
               isTrue=true;
          }
          else{
               isTrue=false;
               break;
          }
     } while(a<b);
     return isTrue;
}
```

```
int main()
{
    int n;
    string word;
    bool isTrue;

    cout << "Enter a word: ";
    cin >> word;

    n = word.length()-1;
     int i = 0;
     raya = myCode(word, i, n);

     if (isTrue==true)
          cout << "\nThe word \"" << word << "\" IS a word
we look for.";
     else
          cout << "\nThe word \"" << word << "\" IS NOT a
word we look for.";

    return 0;
}
```

(a) Identify the purpose of the above program

(2 marks)

(b) Modify the above function into a recursive function.

(5 marks)

5. Analyse the following lists of nodes for a binary tree:

Preorder: srseponrsudennodomia

Inorder: pnoerssru**S**ednnodoima

Bold alphabet marks the root node of the tree.

(a) Construct the binary tree above.

(5 marks)

(b) Show the post order traversal based on the constructed tree.

(5 marks)