First Semester Examination
2021/2022 Academic Session

February 2022

CPT113/CPM213 – Programming Methodology & Data Structures
(Metodologi Pengaturcaraan & Struktur Data)

Duration : 2 hours
(Masa : 2 jam)

Please ensure that this examination paper contains SEVEN (7) printed pages before you begin the examination.

[Sila pastikan bahawa kertas peperiksaan ini mengandungi Tujuh (7) muka surat yang bercetak sebelum anda memulakan peperiksaan ini.]

Instructions: Answer all **FOUR (4)** questions.

*[**Arahan**: Jawab kesemua **EMPAT (4)** soalan.]*

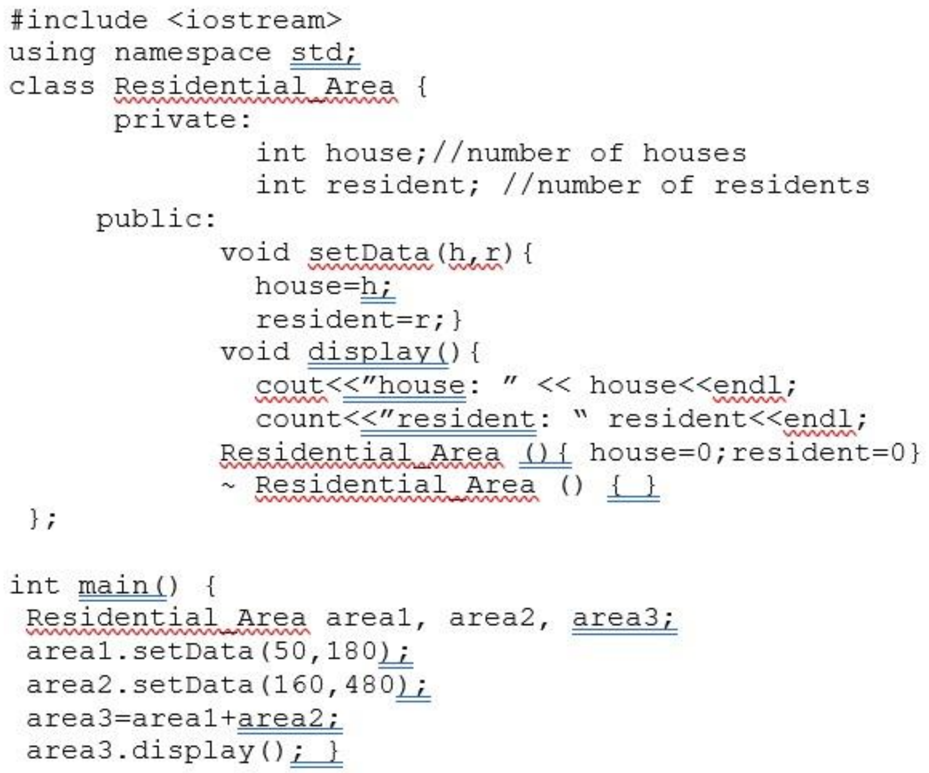
You may answer the questions either in English or in Bahasa Malaysia.

[Anda dibenarkan menjawab soalan sama ada dalam bahasa Inggeris atau bahasa Malaysia.]

1. (a) Suppose that data members are declared as private in the `class A`. Explain two (2) ways that allow C++ statements in `main ()` function to update private data members of `class A.`

(10/100)

(b) Given the following C++ source code:



```
#include <iostream>
using namespace std;
class Residential_Area {
      private:
              int house;//number of houses
              int resident; //number of residents
     public:
            void setData(h,r){
              house=h;
              resident=r;}
            void display(){
              cout<<”house: ” << house<<endl;
              count<<”resident: " resident<<endl;
            Residential_Area (){ house=0;resident=0}
            ~ Residential_Area () { }
 };

int main() {
 Residential_Area area1, area2, area3;
 area1.setData(50,180);
 area2.setData(160,480);
 area3=area1+area2;
 area3.display(); }
```

Analyse the above program by explaining the following:

- the problem arises when the source code is compiled
- the solution to solve the above problem by providing source code

(20/100)

(c) Suppose the class named `MyPoint` that store a point with `x` and `y` coordinates already exists. The class `MyPoint` contains:

- Two (2) private data `x` and `y` to represent the coordinates
- `Default constructor` to set all values with `0`
- A `constructor` that sets a specific coordinate `x` and `y`.
- Two methods that return the value of `x` and `y` respectively.
- A method named `setPoint` to set the value of `x` and `y`

(i) You are required to store a new coordinate value, `z` coordinate to model a point in three-dimensional space. Using class `MyPoint`, define a new class named `threeDPoint` which inherits publicly from the class `MyPoint`. The class `threeDPoint` contain the following:

- One (1) private data `z` to represent the z coordinate
- `Default constructor`
- A `constructor` that sets a point with three specified coordinates
- A method that returns the value of `z`.
- A method named `setPoint` to set values of `x`, `y` and `z`
- A method named `distance` that returns the distance between the two points in the three-dimensional space. This method receives another object as a parameter which contains coordinates of another point.

(20/100)

(ii) Show the following methods:

- A constructor that sets a point with three specified coordinates
- `setPoint` to set values of `x`, `y` and `z`
- `distance` that returns the distance between the two points in the three-dimensional space. The following is the formula to calculate the distance between the two points:

$$\texttt{Distance} = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2 + (z_1-z_2)^2}$$

Hint: You can use available function **sqrt** and **power**

(30/100)

(iii) Show the correct C++ statements in `main ()` program to do the following:

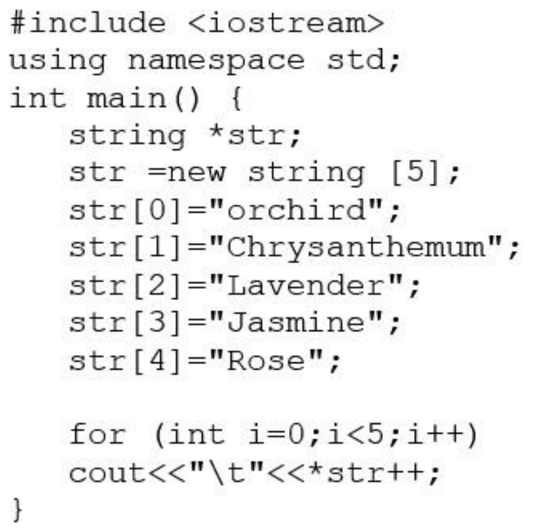
- Declare two arrays object named `side1` and `side2` to store coordinate of 4 points respectively
- Set the following coordinates for `side1`:
  `(2.5,3,5),(12,15,20),10,30,20),(20.5,30.5,40)`
- Set the following coordinates for `side2`:
  `(10,14.5,25), (6,7.5,12), (13,21,18),(56,11.5,13)`
- Display the distance between the two points in `side1` and `side2` respectively. E.g., the distance between the points `side1[0]` and `side2[0]`.

(20/100)

2. (a) Explain one (1) advantage of dynamic memory.

(10/100)

(b) (i) Given the following C++ source code:



```
#include <iostream>
using namespace std;
int main() {
   string *str;
   str =new string [5];
   str[0]="orchird";
   str[1]="Chrysanthemum";
   str[2]="Lavender";
   str[3]="Jasmine";
   str[4]="Rose";

   for (int i=0;i<5;i++)
   cout<<"\t"<<*str++;
}
```

Analyse the above C++ source code for the following:

- What will happen when the program terminates?
- Provide the solution to solve the above problem.

(20/100)

(ii) Given the following C++ source code:

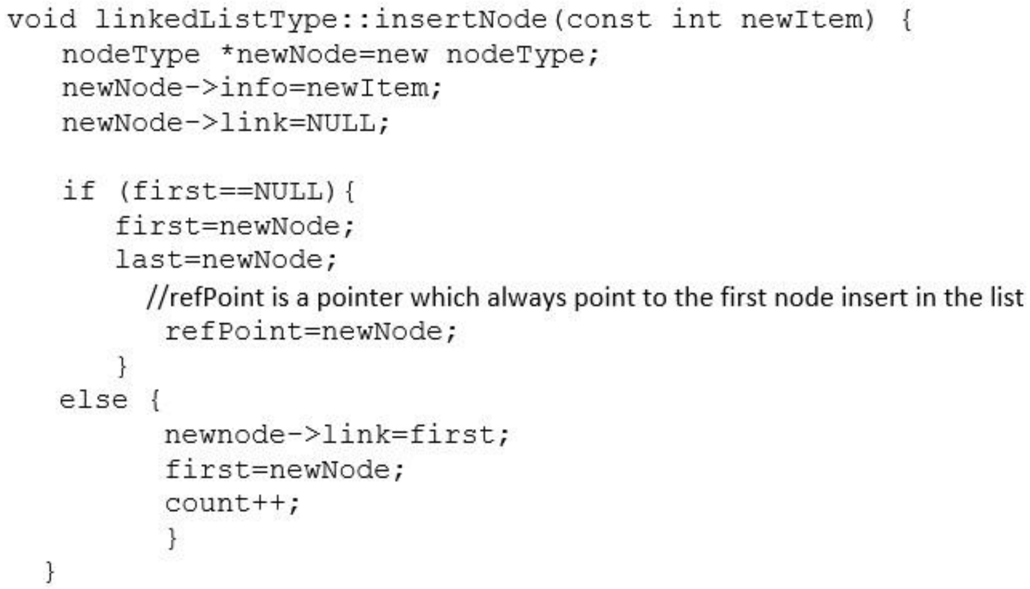
```
nodeType *newNode,*current, *trailCurrent;
…
…
//node to be deleted is in the middle of the list or
  at the end of the list
if(found){ //if node to be deleted found
   current->link=trailCurrent;
   delete current;
   count--;
   }
```

Analyse the above C++ source code for the following.

- Explain the problem that occurs with the above code
- Show the correct source code.

(20/100)

2. (c) Given the following source code:



```
void linkedListType::insertNode(const int newItem) {
   nodeType *newNode=new nodeType;
   newNode->info=newItem;
   newNode->link=NULL;

   if (first==NULL){
      first=newNode;
      last=newNode;
        //refPoint is a pointer which always point to the first node insert in the list
        refPoint=newNode;
      }
   else {
         newnode->link=first;
         first=newNode;
         count++;
         }
  }
```

Modify the above code, so that if the new item is bigger than the value in the node pointed by the pointer `refPoint`, add new item at the end of the linked list. Otherwise add new item at the front of the linked list.

(30/100)

(d) Compare the advantage of doubly linked list over single linked list by giving a reason to perform the following operations:

- Delete node in the middle of list
- Display data in the list

(20/00)

3. (a). Given the abstract data type (ADT) for dynamic stack template in Stack.h below,

```
template <class Type>
class linkedStackType
{
   private:
   struct nodeType
      {
         Type info;
         nodeType *link;
      };
      nodeType *stackTop;
   void copyStack(const linkedStackType<Type>& otherStack);

   public:
   const linkedStackType<Type>& operator=
   (const linkedStackType<Type>&);
   bool isEmptyStack() const;
   bool isFullStack() const;
   void initializeStack();
   void push(const Type& newItem);
    void inStack(const Type& newItem);
   Type top() const;
   void pop();
   linkedStackType();
   linkedStackType(const linkedStackType<Type>& otherStack);
   ~linkedStackType();
};
```

(i). Show a method `inStack` that could transform a stack as in Figure 1(a) to the form as in Figure 1(b).

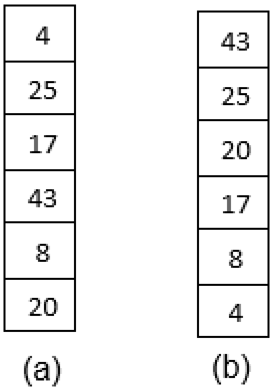


Figure 1

(40/100)

		(ii).	Show the complete `C++ main` program to enter the numbers `25, 35, 5` and `20` into a stack in descending order using `Stack.h` as ADT and display the content of the stack. (30/100)
	(b).	Describe the purpose of the following queue operations along with their definitions	
		(i).	`isEmptyQueue` (5/100)
		(ii).	`Front` (5/100)
		(iii).	`addQueue` (5/100)
	(c).	Explain the changes needed in C++ statements for `addQueue` and `deleteQueue` operations when an array-based circular queue is used. (15/100)	
4.	(a).	Computers understands human instructions using low-level machine codes. Machine codes are actually the binary representations of the instructions. Given any instructions, computers will transform this value into binary before it is able to execute them.	
		(i).	Infer a recursive algorithm definition to perform the above given any decimal numbers. (40/100)
		(ii).	Show a complete C++ program that could implement the process in Question 4(a)(i). (30/100)
	(b).	Draw a binary tree based on the traversal order sequence given below: Inorder – 4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90 Preorder – 25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90 Postorder – 4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25 (30/100)	

- oooOooo -