First Semester Examination
2019/2020 Academic Session

December 2019 / January 2020

CPT113 – 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 **THREE (3)** questions.

*[**Arahan**: Jawab kesemua **TIGA (3)** 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.]

In the event of any discrepancies, the English version shall be used.

[Sekiranya terdapat sebarang percanggahan pada soalan peperiksaan, versi bahasa Inggeris hendaklah diguna pakai.]

(1). (a). (i). Explain the **one** (1) feature of object oriented programming with one example.

(ii). Why constructor is important. Give **two** (2) types of constructors.

(iii). Give **two** (**2**) advantages of friend functions.

(25/100)

(b). Consider the following declarations:

```
Class xClass
{
public:
   void func();
   void print() const;
   xclass();
   xclass (int, double);
private:
   int u;
   double w;
};
 xClass ;
```

(i). How many members does class `xClass` have?

(ii). How many private data members does class `xClass` have?

(iii). How many constructor does class `xClass` have?

(iv). Write the definition of the member function `func` so that `u` is set to 10 and `w` is set to 15.3.

(v). Write the definition of the member function `print` that prints the contents of `u` and `w`.

(vi). Write definition of the default constructor of the class `xClass` so that the private data members are initialized to 0.

(30/100)

(c). (i). Give a definition for class `Square` that is a derived class of the base class `boxType` is given below. This class should have an additional data member `notEqual` of type `bool`, a function `isEqual` with no parameters and returns a value of type `bool`, and suitable constructors.

```
class boxType
{
public:
   boxType();
   void printAnswer();
protected:
   int length;
   int width;
};
```

(30/100)

(ii). Construct the definition of function `isEqual`.This function will return thru if the length is equal to width otherwise it will return false.

(15/100)

(2). The following diagram represents nodes which stores student's scores in ascending order. Using the diagram in Figure 1, answer the following questions.

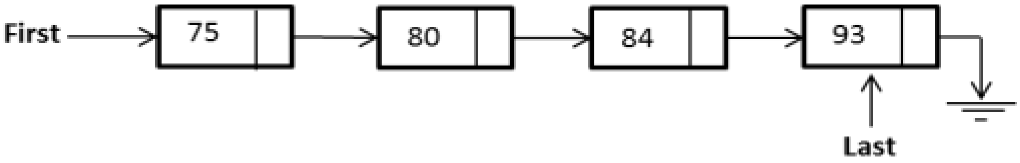

Figure 1
Rajah 1

(a). Complete the following `struct` definition:

```
template <class Type>
struct studentScore
{
(i)  __________ // to store data
(ii) __________ // to declare pointer
}
```

(5/100)

(b). The following is the incomplete abstract class definition as ADT for the linked List Figure 1:

```
Template <class Type>
    class ScoreType {
       public:
              const ListType<Type> & operator= (const
              listType<Type>&);
              bool isEmptyList() const;
              void display();
              int length() const;
              void destroyList();
              Type front() const;
              (i)  void insert_inOrder(______);
              void deleteNode (Type& deleteItem);
              (ii)  __________ // default constructor
              (iii) _________ //default destructor

        protected:
              (iv)  __________ //to declare pointer named first
              (v)   __________ // to declared pointer named last
              };
```

(i). Write the incomplete statements for number (i),(ii),(iii),(iv) and (v) above.

(15/100)

(ii). Show the UML class diagram of the class `ScoreType`

(5/100)

(iii). Write a complete function `display()` to display all data in the linked list.

(5/100)

(iv). Write a complete function `insert_inOrder()` to insert data in ascending order into the linked list. You must consider the following cases:

- Case 1 The list is initially empty
- Case 2 The new item is smaller than the smallest item in the list
- Case 3 The item to be inserted somewhere in the list
- Case 4 The new item is larger than all item in the list

(35/100)

(c). Given the following program:

```
#include <iostream>
using namespace std;
int fun(int a, int b)
{
   if (b == 0)
       return 0;
   if (b % 2 == 0)
       return fun(a+a, b/2);
   else
       return fun(a+a, b/2) + a;
}

int main()
{
  cout<< "The result is: " << fun(4,3) << endl;
  return 0;
}
```

(i). State the base and general cases.

(ii). What is the output of following program?

(iii). What does the function `fun()` do in general

(35/100)

(3). (a). Evaluate the following postfix expression. Show the status of stack after execution of each operation separately:

```
2  13  +  5  - 6 3 / 5 * <
```

(10/100)

(b). Consider the following statements

```
stackType <int> stack ;
int x ;
```

Suppose that the input is

```
14 45 34 23 10 5 -999
```

Show the output of the following segment of code.

```
stack.push(5) ;
cin>>x ;
while (x != -999)
{
     if (x % 2 ==0)
   {
     if (!stack.isFullStack( ))
         stack.push(x) ;
   }
   else
        cout<<‘x= ‘<<x<< endl ;
  cin>>x ;
}
cout <<‘Stack Element :  ‘;
while(!stack.isEmptyStack( ))
{
     cout<<‘  ‘<<stack.top( ) ;
     stack.pop( );
}
cout<<endl ;
```

(25/100)

(c). (i). Build the Binary-Search-Tree T (BST) using the following sequence of numbers starting with an empty tree.

```
60 50 70 30 53 35 57 32 40 48 45 80 75 77
```

(10/100)

(ii). Based on Binary Search Tree T build in 3c(i) , redraw the BST after perform the following sequence of operations:

- Delete node 30
- Delete node 45

(10/100)

(iii). Assume the following definition of function insertNode as follows:

```
void BinaryInsertTree:: insertNode ( const Nodetype
&insertNode)
nodetype * current ;      //pointer to traverse the tree
nodetype *trailcurrent;  // pointer behind current
nodetype *newnode;       //  pointer to create the node
```

Write a C++ function to insert a new node 85 into a Binary Search Tree T in 3(C(ii)).

(30/100)

(iv). Using BST in 3(iii), list the node numbers for traversing the tree using the

- preorder traversal method.
- inorder traversal method.
- postorder traversal method.

(15/100)

- oooOooo -