MULTIMEDIA UNIVERSITY

STUDENT ID NO

TABLE NO: _____

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2015/2016

TCP1201 – OBJECT-ORIENTED PROGRAMMING AND DATA STRUCTURES

(All sections / Groups)

2 MAR 2016
2:30 pm – 4:30 pm
(2 Hours)

Question	Mark
1	
2	
3	
4	
Total	

INSTRUCTIONS TO STUDENT

1. This question paper consists of 11 printed pages (including the cover page) with four questions. Answer all questions.
2. Attempt all questions. Each question carries 10 marks.
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) You are assigned to create a simple system for a Wild Monkey Zoo to keep track of the monkeys kept in the zoo.

i) Implement a class Monkey to store the details of a particular monkey species in the zoo. The Monkey class has a species name, a country of origin and the number of monkeys of this species that are kept in the zoo. Provide the following functions for the Monkey class:
- One constructor with parameters to initialize all the attributes of the class.
- Three query functions that return each of the three attributes of the class.
- One update function that allows the zookeeper to change the number of monkeys for that species.

[3]

```
class Monkey
{









};
```

ii) Provide the class declaration for a Zoo class. The Zoo class stores the information of all the monkey species kept in the zoo and has the following functions:
- An add function that allows a new species to be added to the zoo.
- A search function that determines whether a species of monkey is kept in the zoo.
- A print function that displays the details of all the monkey species in the zoo.

(Note: You do not need to provide the implementation for all the above functions)

[2]

```
class Zoo
{



```

```
};
```

iii) Implement the search function of the Zoo class. The search function will return true if the particular target species is found in the zoo, and false otherwise.

[2]

(b) An exception handler is code that handles an exception (runtime error) when it occurs. One of the advantages of an exception handler is it can provide clarity on the section of code that handles the error. Convert the following code segment to use an exception class to handle the invalid index error.

```
int main()
{
    int arrayLength = 3;
    int anArray[arrayLength];

    int index = 5;
    int nValue = anArray[index];
    string error = "Invalid index";

    if (index < 0 || index >= arrayLength)
        cout << "An array exception occurred (" << error
             << ")" << endl;
```

```
        else
            nValue++;
        return 0;
    }
```

[3]

QUESTION 2 [10 marks]

(a) Explain the difference between overriding a member function in a derived object class and function overloading.

[2]

(b) The following code excerpt contains a class declaration called RM and a driver program for use in the question that follows below.

```
class RM : public Money
{
   private:
      int ringgit, sen;
   public:
      RM() {
            ringgit = 0;
            sen = 0;
      }
      void virtual print() {
         cout << "RM" << ringgit << "." << sen << endl;
      }
};

int main() {
   RM acct1, acct2;
   acct1 = 2.99; // save separately into ringgit and sen
   acct2 = 3.50; // save separately into ringgit and sen
   cout << "Amount in acct1 = " << acct1 << endl;
   cout << "Amount in acct2 = " << acct2 << endl;
   cout << "acct1 + acct2 = "<< acct1+acct2 << endl;
   return 0;
}
```

i) Explain how the RM class declaration above shows an implementation of polymorphism.

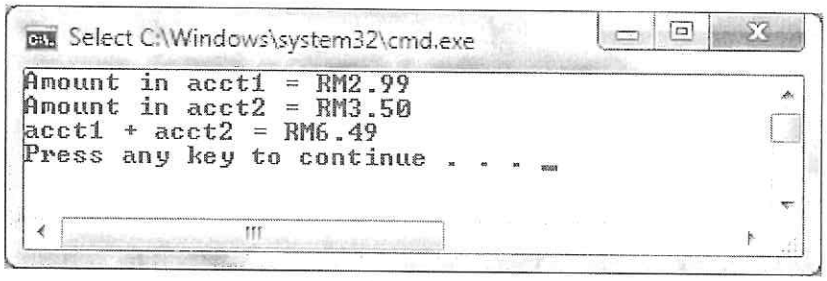
[1]

ii) Complete the class by adding class operators for assignment (=) and addition (+) based on the required actions in the `main()` function shown above (note the

assignment function takes in a double data type and separates the value into two parts for the ringgit and the sen attributes)

[5]

iii) To complete the class, overload the << operator such that it works to display the following when the program is compiled and executed.



[2]

```
friend ostream &operator<<(                    , const              )
{




}
```

QUESTION 3 [10 marks]

(a) Given below is an incomplete code excerpt of a Linked List implementation. Complete the `insertAfter` method. The method should insert a node containing `newElement` after a node that contains `target` data.

[3]

```
//Filename: Node.h
#ifndef _NODE
#define _NODE
template<class ItemType>
class Node
{
private:
   ItemType         item; // A data item
   Node<ItemType>* next; // Pointer to the next node

public:
   Node();
   Node(const ItemType& anItem);
   Node(const ItemType& anItem, Node<ItemType>* nextNodePtr);

   // The usual get and set functions below for private data members
   void setItem(const ItemType& anItem);
   void setNext(Node<ItemType>* nextNodePtr);
   ItemType getItem() const ;
   Node<ItemType>* getNext() const ;
};
#endif
```

```
//Filename: LinkedList.cpp
#include "Node.h"
#include <cstddef>
template <class T>
class LinkedList
{
private:
  Node<T> *start;
  ...             // other data members

public:
  ...              // other member functions

  bool insertAfter (T& target, T& newElement)
  {
    // New node creation
    Node<T>* newNode = _____;
    newNode->setItem(_____);
    bool found = false;


    // Node traversal
    Node<T>* ptr = start;
    while (_____&& !found)
```

```
    {
      if (ptr->getItem() == target)
        found = true;
      if (!found)
        ptr = _____;
    }


   // Node insertion
    if (found) {
      newNode->setNext( _____);
      ptr->setNext( _____);
      return true;
    }
  }

};
```

(b) Explain one benefit of link-based list implementation over array-based list implementation.

[2]

(c) Give two uses of ADT stacks.

[2]

(d) What is the relationship between recursion and stacks?

[1]

(e) Complete the method definition `display()` for the following array-based implementation of ADT queue. The method `display()` is used to show all items from a queue.

[2]

```
const int MAX_QUEUE = 50;
```

```
template<class ItemType>
class ArrayQueue
{
private:
    ItemType items[MAX_QUEUE]; // Array of queue items
    int front; // Index to front of queue
    int back; // Index to back of queue
    int count; // Number of items currently in the queue
public:
    ArrayQueue();
    bool  isEmpty() const;
    bool  enqueue(const ItemType& newEntry);
    bool  dequeue();
    /** @throw PrecondViolatedExcep if queue is empty. */
    ItemType peekFront() const throw(PrecondViolatedExcep);
    void display() const;
    bool clear();
}; // end ArrayQueue

template<typename ItemType>
void ArrayQueue<ItemType>::display() const
{
   // begin code




  // end code

}
```

QUESTION 4 [10 marks]

(a) Write a recursive function to calculate the sum of all integers from 1 to n.
e.g:

```
sum(3) = 1+2+3 = 6
sum(4) = 1+2+3+4 = 10
```

[2]

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

// Begin code




// End code
```

```
int main(void)
{
  int n;
  cin >> n;
  cout << "sum(" << n << ") ="  << sum(n) << endl;
  return 0;
}
```

(b) Below is the code to implement the `getNumberOfNodes()` member function of a binary tree. This function counts the number of nodes inside a binary tree. Fill in the blanks to complete the code.

[4]

```
template<class ItemType>
class BinaryNode
{
private:
   ItemType              item;          // Data portion
   BinaryNode<ItemType>* leftChildPtr;  // Pointer to left child
   BinaryNode<ItemType>* rightChildPtr; // Pointer to right child

public:
   ...
   BinaryNode<ItemType>* getLeftChildPtr() const;  // Returns leftChildPtr
   BinaryNode<ItemType>* getRightChildPtr() const; // Returns rightChildPtr
};


template<class ItemType>
class BinaryNodeTree
{
private:
   BinaryNode<ItemType>* rootPtr;

protected:
   //------------------------------------------------------------
   // Protected Utility Methods Section:
   // Recursive helper methods for the public methods.
   //------------------------------------------------------------

   int getNumberOfNodesHelper(BinaryNode<ItemType>* subTreePtr) const;

public:
   ...
   int getNumberOfNodes() const;
   ...
};


template<class ItemType>
int BinaryNodeTree<ItemType>::getNumberOfNodes() const
{
   return getNumberOfNodesHelper( _____________________);
}

template<class ItemType>
int BinaryNodeTree<ItemType>::getNumberOfNodesHelper(BinaryNode<ItemType>*
subTreePtr) const
{
```

```
    return ____+
    getNumberOfNodesHelper( _____ ) +
    getNumberOfNodesHelper( _____ );
}
```

(c) The following numbers are inserted one after another to construct a binary search tree.

5, 9, 1, 8, 6, 2, 4

Draw the resulting binary search tree.

[2]

(d) What is the output the following program?

```
#include <iostream>
#include <set>
using namespace std;
int main()
{
  set<int> s;
  s.insert (81);  s.insert (9);
  s.insert (23);  s.insert (14);  s.insert (81);

 set<int>::iterator it;
  it = s.begin();
  while (it != s.end())
  {
    cout << *it++ << " ";
  }
  cout << endl;

  int target = 23;
  it = s.find (target);
  it--;
  cout << *it << endl;
  return 0;
}
```

[2]

End of Page