Program Design
Final Test

Closed Book 6/21/2016

I. Matching (25 %)

____1. Inheritance
____2. Derived class
____3. "Has a" relationship
____4. "Is a" relationship
____5. Single inheritance
____6. Base class
____7. Indirect base class
____8. Base-class initializer
____9. Multiple inheritance
____10. dynamic binding
____11. abstract base class
____12. polymorphism
____13. virtual function
____14. pure virtual function
____15. 1. catch block
____16. virtual base-class destructor
____17. Exception handling
____18. catch(...)
____19. bad_alloc
____20. try block
____21. throw()
____22. virtual function table
____23. Abstract base class
____ 24. Concrete class
____25. typeid

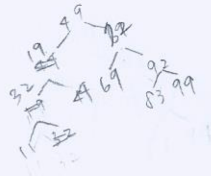
a. Class that is defined, but never intended to be used by the programmer to create objects.
b. Part of C++'s run-time type information.
c. An executing program uses this to select the proper function implementation each time a virtual function is called.
d. Ensures proper cleanup when processing dynamically allocated objects in a class hierarchy, polymorphically.
e. Class from which objects can be instantiated.
f. Class from which others are derived.
g. Deriving from more than one base class.
h. Class that is created by inheriting from an existing class.
i. Inheritance.
j. Passes arguments to the base-class constructor.
k. Base class that is not listed explicitly in the derived class's definition.
l. Composition.
m. Deriving from only one base class.
n. New classes are created from existing classes.
o. Class that is defined, but never intended to be used by the programmer to create objects.
p. Function prototypes that end with "= 0."
q. Allows objects of different classes related by inheritance to respond differently to the same message.
r. Encloses the code that may generate an exception.
s. Programming "in the general."
t. Occurs only off pointer or reference handles.
u. Helps improve a program's fault tolerance.
v. Encloses the code that may generate an exception.
w. Exception thrown when new fails.
x. Indicates that a function does not throw exceptions.
y. "Catch all" handler that catches any exception.

II. Closing (15 %)

a. A self-referential class is used to form dynamic data structures that can grow and shrink at execution time.
b. The new operator is used to dynamically allocate memory and construct an object; this operator returns a pointer to the object.
c. The pointer to the next node in a linked list is referred to as a(n) link.
d. The delete operator is used to destroy an object and release dynamically allocated memory.
e. A(n) tree is a nonlinear, two-dimensional data structure that contains nodes with two or more links.
f. The nodes of a(n) binary tree contain two link members.
g. A tree node that has no children is called a(n) leaf node.
h. The four common traversal algorithms for binary search trees are inorder, preorder, postorder and 階層.
i. A queue is referred to as a(n) FIFO data structure, because the first nodes inserted are the first nodes removed.
j. A stack is referred to as a(n) FILO data structure, because the last node inserted is the first node removed.
k. Each link in a tree node points to a(n) ____ or ____ of that node.
child

III. Definition of a **List** class using a node class **Node** is as: (20%)

```
--------------------------------------------------------
class List
{ public:
    List(); // constructor
    ~List(); // destructor
    void insertAtFront( const Node & );
    void insertAtBack( const Node & );
    bool removeFromFront(Node & );
    bool removeFromBack(Node & );
    bool isEmpty() const;
    void print() const;
  private:
    Node *firstPtr; // pointer to first node
    Node *lastPtr; // pointer to last node
    // utility function to allocate new node
    Node *getNewNode( const Node & );
}; // end class List
--------------------------------------------------------
```



The definition of **Queue (FIFO)** and **Stack (FILO)** based on the list definition can be done by using both *inheritance* and *composition*. Please give these **four** definitions respectively.

IV. Suppose there are integers generated and input to your program by the order as: (Total 30%)

11, 19, 32, 44, 49, 69, 72, 83, 92, 99

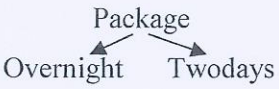
a. Please manually draw the binary search tree created by using these integers. *(5 %)*

b. Please manually traversal the binary search tree in **inorder**, **preorder**, and **postorder**. *(5 %)*

c. **Please** define the class **TreeNode** and the class **Tree** with proper data. In the **Tree** class define and implement a **insertnode** methed, which can be used to insert a new node into a binary search tree, and the three common traversal methods. **(Do not use template class!)** *(10%)*

d. Please define and implement a search method **binTreeSearch** that can search a particular integer in the binary search tree. For the particular binary search tree created by **c.**, what is the performance difference between the **binTreeSearch** and a **linear search** method for the integers that are stored in a linear array. *(10 %)*

V. Define (without implementation) the **Package** classes hierarchy shown as below.



All the classes should have a method **CalculateCost()** to calculate the cost of the packages. Define and inherite the classes so that they can demonstrate the ***polymorphism*** when calling the **CalculateCost()** method. Create a program segment that uses a vector of **Package** pointers to objects of each subclass in the hierarchy. Create the objects of a Overnight and a Twoday classes and put them into the vector. Then create a loop to print out the costs of all the packages in the vector by calling their **CalculateCost()** method. (ignore other irrelvant methods) **(10%)**