2018/06/26

Program Design Final-Test – 2018

Open for the second part

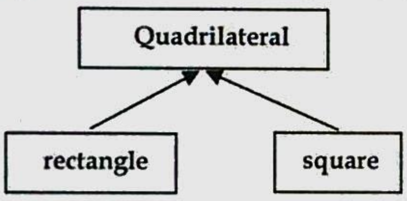
Name: **Student NO.**

III. Definition of a **List** class using a node class **Node** is as follows: **(10%)**

```
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 definitions of **Stack (FIFO)** and **Queue (FIFO)** based on the list definition above can be done by using *composition*. Please implement the definitions.

IV. An inheritance hierarchy for classes **Quadrilateral**, **Rectangle** and **Square** uses **Quadrilateral** as the base shape class of the hierarchy as follows:



a. Design the **Quadrilateral** as an abstract base class with no data member, from which the two shape classes **Square (with data members: h,w)** and **Rectangle (with data members: l)** are derived and defined with proper data members. The **Quadrilateral** should makes a member function *calculatearea()* abstract. Each derived class should override function *calculatearea()* to calculate and return the area of the shape. **Define the hierarchy so that you can use polymorphism later. (include only relevant data members and member functions only.)** **(10%)**

b. Modify the **binary tree node** and binary search tree definitions (you can find it in Fig20.20 and 20.21) to create binary search tree with Quadrilateral objects as tree nodes that contain a data of pointers pointing to objects of **rectangle** or **square** class (**do not use templates**). Define a member function *BinTreeInsert()* that inserts Quadrilateral objects into a binary search tree according to the **area** of the object (calculated by the member function *calculatearea()* of the object, be aware to use **polymorphism** here). You have to include the implementation part of the function only in the class. **(15%)**

c. Assume there are **twelf** objects (sixes for each derived class, and use only integer lengths) with their data originally saved in a two dimentional array **rect[6][6]** and a one dimentional array **squr[6]**. Use the following lengths and widths for **rect**: 4x2, 3x2, 8x4, 9x3, 6x4, 5x2, and 3, 4, 5, 8, 6, 2 for **squr**. Create two loops, one for each array, that read data from the arrays and call the constructors of the derived classes to create objects and insert them into a **binary search tree with Quadrilateral** nodes by the member function *BinTreeInsert()* defined in b. according to their areas. **(10 %)**

d. Please manually draw the binary search tree containing the twelf objects using nodes like 5x2(10), 4x4(16), and 9x3 etc. **(5 %)**

e. Modify the binary search tree **inorder** traversal member function (**no template**) so that it prints out all nodes information like:

rectangle (5x2) with area 10
square (4x4) with area 16
……

Note you have to determine if a node points to a rectangle or a square. (*hint*: use run time type information.) **(10 %)**

f. Define a member function *BinTreeSearch()* for the binary search tree, which inputs an integer and check if it is the area of an object in the binary search tree you created in c. (Calculate the area of an object before comparing it with the input integer) **(10 %)**

Enjoy your examinations!

Don't cry babe. It's just a movie...
Don't cry babe. It's just football...

NOTE: Next Exercise: Combine 20.12 and 20.13 to evaluate an expression. Limite the operators just to +, -, *, /. You can download the header files from the web page. The zip file contains several files. You have to finish the file: Ex21_12_ToBeFinished (Due in 15/Jul.)