1. **(5 marks)** Implement default constructor, destructor and assignment operator for the following `RestaurantMenu` class.

```cpp
class RestaurantMenu {
public:
    RestaurantMenu(); // implement this one
    RestaurantMenu(vector<double>*, double);
    RestaurantMenu(const RestaurantMenu&);
    RestaurantMenu& operator=(const RestaurantMenu&);
    // implement this one
    ∼RestaurantMenu(); // implement this one
    // other public functions
private:
    vector<double>* menuPrices;
    double taxRate;
}
```

Answers:

```cpp
RestaurantMenu::RestaurantMenu():taxRate(0.0),menuPrices(NULL){}
RestaurantMenu& RestaurantMenu::operator=(const RestaurantMenu& rhs)
{
    if(&rhs != this)
    {
        taxRate=rhs.taxRate;
        delete menuPrices;
        menuPrices=new vector<double>;
        if(rhs.menuPrices==NULL)
        {
            menuPrices = NULL;
        return *this;
```
int i=0;
while(i < rhs.prices->size())
    menuPrices->push_back(rhs.prices->at(i++));
}
return *this;
}
RestaurantMenu::~RestaurantMenu() { delete menuPrices; }

2. (10 marks) Implement the operator== function for the RestaurantMenu class. The function bool RestaurantMenu::operator==(RestaurantMenu& other) returns true if the difference between the average menu price of the restaurant and the average menu price of the other restaurant is less than $2 and the difference between their tax rates is less than 1%. Otherwise, the function returns false.

bool RestaurantMenu::operator==(RestaurantMenu& other)
{
    if (menuPrices == NULL || other.menuPrices == NULL) return false;
    double this_total=0.0;
    int this_i=0;
    while(this_i < menuPrices->size())
        this_total+=menuPrices->at(this_i++);
    double that_total=0.0;
    int that_i=0;
    while(that_i < other.menuPrices->size())
        that_total+=other.menuPrices->at(that_i++);
    return abs(this_total/this_i - that_total/that_i) < 2 && abs(taxRate-
    other.taxRate) < 0.01;
}

3. (4 marks) Are the following 2 constructors operationally equivalent? I.e. do they take the same steps to create a Book object? Justify your answer.

Book: Book(string title_in, int year_in): title(title_in), year(year_in){}
Book: Book(string title_in, int year_in){title=title_in; year=year_in;}

Answer:
No. The first constructor initializes the title and year of a Book object at step 3 using the initialization list. The second constructor initializes the title and year of a Book object at step 4 when the body of the constructor function is executed.

4. (2 marks) Does the following swapXY function swap the x and y locations of a given Point object p? If not, modify the function so that p has its x and y swapped after the following function call: Point p(3.0,2.0); swapXY(p);
void swapXY(Point p)
{
    double temp=p.getX();
    p.setX(p.getY());
    p.setY(temp);
    return;
}

class Point{
public:
    Point(double x_in, double y_in):
        x(x_in), y(y_in) {}
    double getX(){return x;}
    double getY(){return y;}
    void setX(double x_in){x=x_in;}
    void setY(double y_in){y=y_in;}
private:
    double x;
    double y;
}

Answer:

No, the swapXY does not swap the x and y locations of the Point object argument. To make the swap happen, we have to change the swapXY function declaration to void swapXY(Point & p). The body of the swapXY function remains the same.

5. (3 marks) Based on the Point class definition in Question 4, what are the outputs of the following code fragments?

```cpp
int main()
{
    Point p1(2.0,3.0);
    Point p2=p1;
    cout << p2.getX() << "\," << p2.getY() << endl;
    p2.setX(3.0);
    p2.setY(2.0);
    cout << p1.getX() << "\," << p1.getY() << endl;
    cout << p2.getX() << "\," << p2.getY() << endl;
}
```

Answer:

2.0, 3.0
2.0, 3.0
3.0, 2.0

6. (10 marks) The following DeliveryMenu class is a derived class of the RestaurantMenu class in Question 1. Due to operational cost, a restaurant charges different price for special() in delivery menu and in restaurant sit-in menu. Assume we have added a virtual double RestaurantMenu::special() function to the RestaurantMenu class. Implement a function double cheapestSpecialInTown(vector<RestaurantMenu*>) which returns the lowest special price in the given vector of restaurant menus.

```cpp
class DeliveryMenu: public RestaurantMenu {
public:
    double special();
    // other public functions
```
private:
    double deliveryCharge;
}

Answer:

double cheapestSpecialInTown(vector<RestaurantMenu*> menus)
{
    double cheapest=menus.at(0)->special();
    for (int i=1; i < menus.size(); i++)
    {
        if (menus.at(i)->special() < cheapest)
            cheapest=menus.at(i)->special();
    }
    return cheapest;
}

7. (16 marks) In addition to constructors and assignment operator given in Question 1, the following public functions have been defined for RestaurantMenu and DeliveryMenu classes.

```cpp
class RestaurantMenu
class DeliveryMenu:public RestaurantMenu
{
public:
    void f(int);
    int f(string);
    virtual double special();
}
```

In the following table, provide the function called for each of the C++ statements. You should give `ClassName::FunctionName(Parameter Types)` for the called function. You should also indicate if the function binding is determined at compile time (static) or at run time (dynamic).
<table>
<thead>
<tr>
<th>C++ code</th>
<th>function called</th>
<th>static/dynamic binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>//create vector&lt;double&gt; priceList1; RestaurantMenu Chess(&amp;priceList1,0.5);</td>
<td>RestaurantMenu::RestaurantMenu</td>
<td>static</td>
</tr>
<tr>
<td></td>
<td>(vector&lt;double&gt;* , double)</td>
<td></td>
</tr>
<tr>
<td>DeliveryMenu campusPizza;</td>
<td>DeliveryMenu::DeliveryMenu()</td>
<td>static</td>
</tr>
<tr>
<td>RestaurantMenu a(campusPizza);</td>
<td>RestaurantMenu::RestaurantMenu</td>
<td>static</td>
</tr>
<tr>
<td></td>
<td>(const RestaurantMenu&amp;)</td>
<td></td>
</tr>
<tr>
<td>a.special();</td>
<td>RestaurantMenu::special()</td>
<td>static</td>
</tr>
<tr>
<td>a.f(2);</td>
<td>RestaurantMenu::f(int)</td>
<td>static</td>
</tr>
<tr>
<td>DeliveryMenu b(campusPizza);</td>
<td>DeliveryMenu::DeliveryMenu</td>
<td>static</td>
</tr>
<tr>
<td></td>
<td>(const DeliveryMenu&amp;)</td>
<td></td>
</tr>
<tr>
<td>b.special();</td>
<td>DeliveryMenu::special()</td>
<td>static</td>
</tr>
<tr>
<td>b.f(&quot;hello&quot;);</td>
<td>DeliveryMenu::f(string)</td>
<td>static</td>
</tr>
<tr>
<td>//create vector&lt;double&gt; priceList2;</td>
<td>DeliveryMenu::DeliveryMenu</td>
<td>static</td>
</tr>
<tr>
<td></td>
<td>(vector&lt;double&gt;* , double),</td>
<td></td>
</tr>
<tr>
<td>DeliveryMenu InAndOut(&amp;priceList2,5.0);</td>
<td>DeliveryMenu::operator=</td>
<td>static</td>
</tr>
<tr>
<td></td>
<td>(const DeliveryMenu&amp;)</td>
<td></td>
</tr>
<tr>
<td>b=InAndOut;</td>
<td>RestaurantMenu::f(int)</td>
<td>static</td>
</tr>
<tr>
<td>RestaurantMenu* rp=&amp;InAndOut;</td>
<td>RestaurantMenu::special()</td>
<td>dynamic</td>
</tr>
<tr>
<td>rp-&gt;f(2);</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rp-&gt;&quot;special()&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rp=&amp;Chess;</td>
<td>RestaurantMenu::special()</td>
<td>dynamic</td>
</tr>
<tr>
<td>rp-&gt;&quot;special()&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rp-&gt;f(2);</td>
<td>RestaurantMenu::f(int)</td>
<td>static</td>
</tr>
<tr>
<td>Point p(3.0,2.0); //defined in Q4</td>
<td>Point::Point(double,double)</td>
<td>static</td>
</tr>
<tr>
<td>swapXY(p); //defined in Q4</td>
<td>swapXY(const Point&amp;)</td>
<td>static</td>
</tr>
</tbody>
</table>