Objective:

- Resolving issues related to pointer as data member.
- Focusing on Object's initialization and resource Allocation/de-allocation issues and issues related to object manipulation.
- And a bit of logic as always to keep your brains working ☺

Challenge: ***Grocery List***

In today's lab, we are going to implement application related to grocery list management, using which, user should be able to store item name, price and count of items. It will help the user to use this list during grocery in a store where user will dynamically check/uncheck items in the grocery list, items finally selected to be bought will go in purchased list.
Have a look at the following class named GroceryList which will be responsible for maintaining both grocery and purchased list.
Detailed sample run and purpose of each member will help you understand the functionality of overall application.

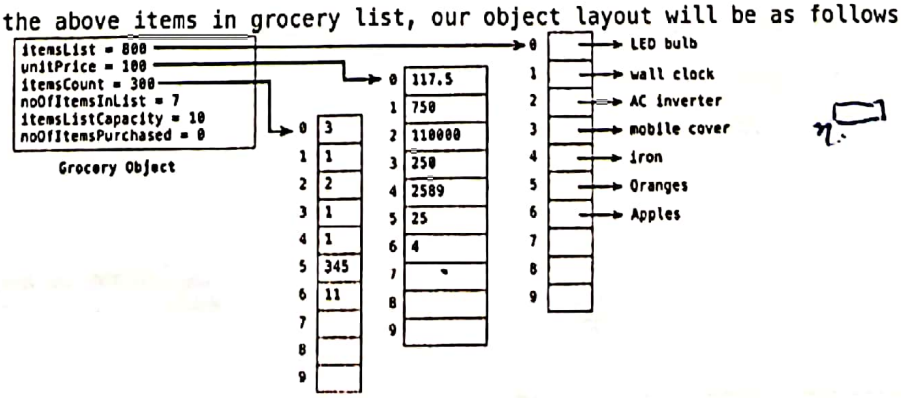
GroceryList data members:

```
int itemsListCapacity;
    //It represents the number of items that can be stored in grocery list.

char * * itemsList;
    //It will point to array of pointers of size itemsListCapacity whose each location will point to a
    char array that represent the item name.

double * unitPrice;
    //It will point to array of double of size itemsListCapacity. unitPrice[i] represent the unit price
    of item at itemsList[i].

int * itemsCount;
    //It will point to array of int of size itemsListCapacity. itemsCount[i] represent the count of
    items of item at itemsList[i].

int noOfItemsInList;
    //It represents the number of items currently stored in the grocery list.

int noOfItemsPurchased;
    //It represents the number of items finally selected to be purchased and put in the shopping cart.
    There is no separate array which will maintain the items purchased rather same arrays itemsList,
    unitPrice, itemsCount will be used to store this information. How? read the explanation below:
```

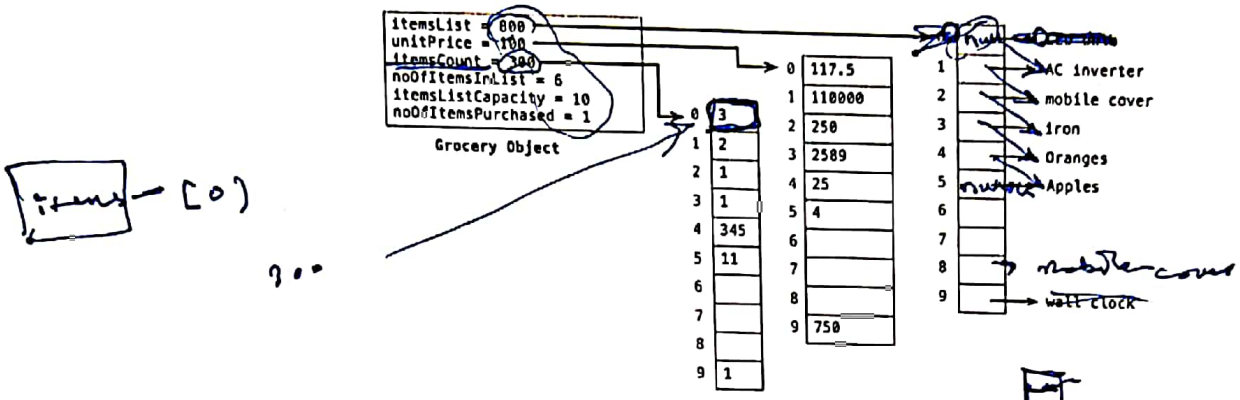
Let's assume that following is our grocery list which we need to add to GroceryList class. So, using a function of GroceryList, we shall add these items in the grocery list.

#	Name	Price	Count	Total
1.	LED bulb	117.5	3	352.5
2.	wall clock	750	1	750
3.	AC inverter	110000	2	220000
4.	mobile cover	250	1	250
5.	iron	2589	1	2589
6.	Oranges	25	345	8625
7.	Apples	4	11	44

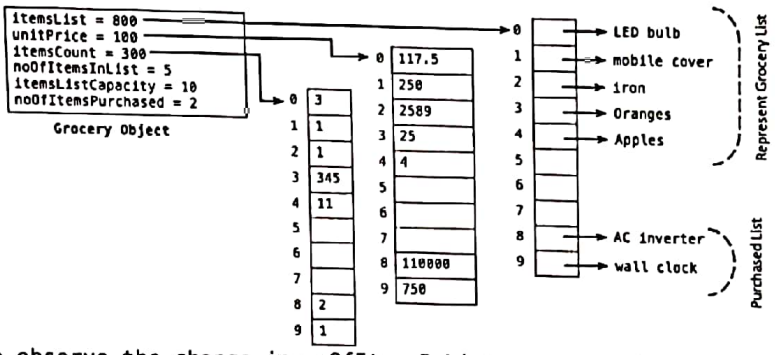
After adding the above items in grocery list, our object layout will be as follows:



So, when user decides to purchase item number 2 (for user, item number 2 means item at index 1 in the array i.e., wall clock). Related function will be called for the said purpose which will move the item in grocery to purchased list (**purchased list start from bottom of *itemsList***). In this case, resultant object layout will be as follows:



Let's assume, user again select item number 2 to be purchased, keep in mind that in the resultant/updated list/object-layout, the item number 2 is now AC inverter stored at index 1 in the itemsList. The resultant layout will be:



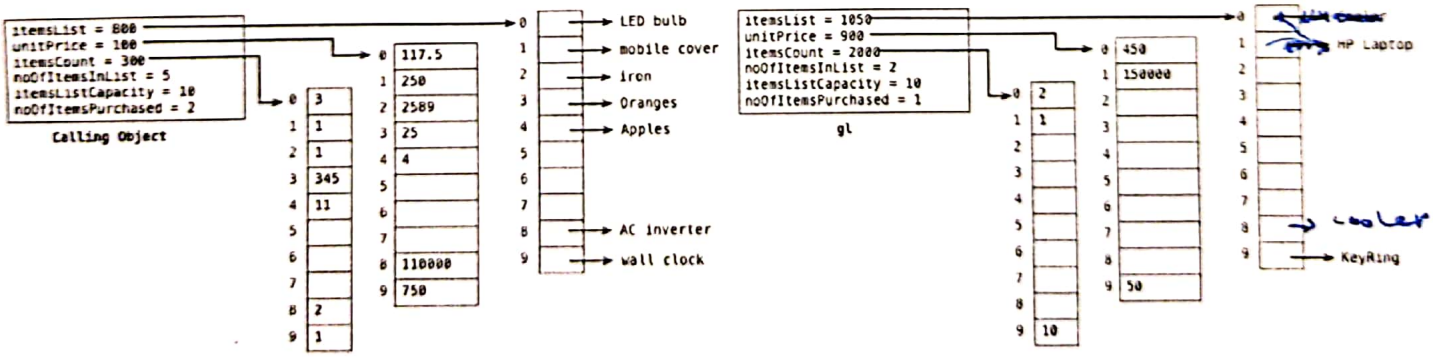
Do observe the change in noOfItemsInList, and noOfItemsPuchased.

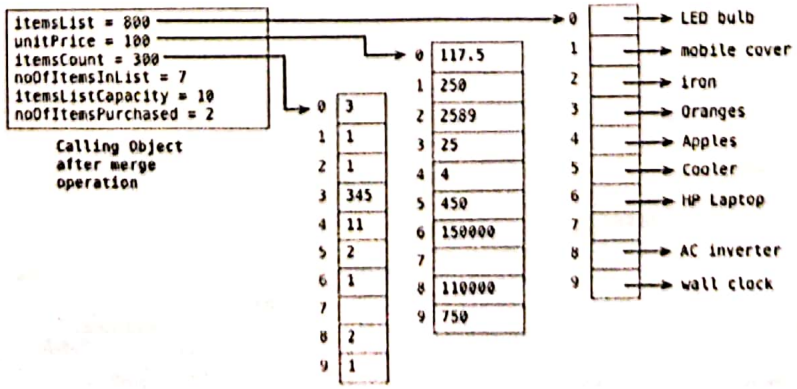
I hope you now understand the concept of purchased list and grocery list and its management in GroceryList class.

Here is the detail of complete GroceryList class.

```
class GroceryList
{
    char * * itemsList;
    double * unitPrice;
    int * itemsCount;

    int noOfItemsInList;
    int itemsListCapacity;

    int noOfItemsPurchased;

    bool isEmpty(); //return true if grocery list is full otherwise false.

    bool isFull(); //return true if grocery list is empty otherwise false.

    void resize(int newSize);
        //resizes the grocery list when gets full. It is recommended to make the new capacity
        double. See the sample run for detail understanding.

public:
        //constructor implementation is already given. No change is allowed in it. The default
        capacity of grocery list is 10 as seen in the code below.
    GroceryList (int cap = 10)
    {
```

```
    if (cap<=0)
        cap = 10;
    itemsListCapacity = cap;
    noOfItemsInList = 0;
    itemsList = new char * [itemsListCapacity];
    unitPrice = new double [itemsListCapacity];
    itemsCount = new int [itemsListCapacity];

    noOfItemsPurchased = 0;
}

GroceryList( const GroceryList & ref ); //creates deep copy of the received object

void addItem (const char * itemName, int itCount, double untPrice);
    //It adds the received item and its details in grocery list.
    //In order to avoid further complexity, we assume that the items received are unique on the
    basis of their name.

void removeItem (int itemNumber);
    //It removes the item from grocery list at the given itemNumber. Reminder: for user, item
    number 1 means item at index 0 in array.

void purchaseItem (int itemNumber);
    //It purchase the item according to given itemNumber and moves it to purchased list already
    elaborated in the diagrams above.

void printGroceryList();
    //It prints the items in the grocery list. See sample run for details.

void printPuchasedList();
    //It prints the items in the purchased list. See sample run for details.

void mergeLists ( GroceryList gl );
    //It adds/merge the grocery list items of received object in the calling object grocery
    list. Make sure that an item in gl, which is already present in *this object must not be
    added in it.
    See the diagram below, which brief that gl object grocery list items are merged into calling
    object. Note: there shouldn't be any change in gl object after this operation. In the
    example given below, resizing was not needed because calling object has enough space to
    accommodate the grocery list items of received object. In case, if resizing needed then you
    need to handle that case too while merging.
```






```
~GroceryList();
    //You know what to do.
};
```

Sample Run – 1: purchased list functionality

```
int main()
{
    GroceryList gList;
    gList.addItem("LED bulb", 3, 117.5);
    gList.addItem("wall clock", 1, 750);
    gList.addItem("AC inverter", 2, 110000);
    gList.addItem("mobile cover", 1, 250);
    gList.addItem("iron", 1, 2589);
    gList.printGroceryList();
    gList.purchaseItem(2);
    gList.printGroceryList();
    gList.purchaseItem(2);
    gList.printGroceryList();
    gList.printPuchasedList();
    return 0;
}
```

Console Output

```
<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..LED bulb............117.5    3      352.5
2..wall clock..........750      1      750
3..AC inverter.........110000   2      220000
4..mobile cover........250      1      250
5..iron................2589     1      2589
Total Amount _____________________________ 223941.50

<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..LED bulb............117.5    3      352.5
2..AC inverter.........110000   2      220000
3..mobile cover........250      1      250
4..iron................2589     1      2589
Total Amount _____________________________ 223191.50

<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..LED bulb............117.5    3      352.5
2..mobile cover........250      1      250
3..iron................2589     1      2589
Total Amount _____________________________ 3191.50

<<<<<<<<<<<<<<<<< Purchased List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..wall clock..........750      1      750
2..AC inverter.........110000   2      220000
Total Amount _____________________________ 220750.00
```

Sample Run – 2: resize feature

```
int main()
{
    GroceryList gList;
    gList.addItem("first", 3, 117.5);
    gList.addItem("second", 1, 750);
    gList.addItem("third", 2, 110000);
    gList.addItem("fourth", 1, 250);
    gList.addItem("fifth", 1, 2589);
    gList.addItem("sixth", 14, 5);
    gList.addItem("seventh", 1, 7);
    gList.addItem("eighth", 3, 67);
    gList.addItem("nineth", 2, 5);
    gList.addItem("tenth", 11, 25);
    gList.printGroceryList();
    gList.printPuchasedList();
```

Console Output

```
<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..first...............117.5    3      352.5
2..second..............750      1      750
3..third...............110000   2      220000
4..fourth..............250      1      250
5..fifth...............2589     1      2589
6..sixth...............5        14     70
7..seventh.............7        1      7
8..eighth..............67       3      201
9..nineth..............5        2      10
10.tenth...............25       11     275
Total Amount _____________________________ 224504.50

<<<<<<<<<<<<<<<<< Purchased List >>>>>>>>>>>>>>>>>
```

```
                                    #  Name                 Price    Count  Total
                                    ================================================
                                    Total Amount _____________________________ 0.00
    gList.addItem("eleventh", 1, 300);
    //will be resized
     //initial capacity of itemList was 10 as
     written in constructor but in addItem, it
     was checked that no room for further
     addition of item in the list so it is
     resized by creating new arrays of double
     size and copied the old data into the new
     arrays alongwith the new item.
```

```
    gList.printGroceryList();
```

```
<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..first.............117.5    3      352.5
2..second............750       1      750
3..third.............110000    2      220000
4..fourth............250       1      250
5..fifth.............2589      1      2589
6..sixth.............5         14     70
7..seventh...........7         1      7
8..eighth............67        3      201
9..nineth............5         2      10
10.tenth.............25        11     275
11.eleventh..........300       1      300
Total Amount _____________________________ 224804.50
```

```
    return 0;
}
```

Sample Run – 3: merge list feature

Console Output

```
int main()
{
    GroceryList gList;
    gList.addItem("LED bulb", 3, 117.5);
    gList.addItem("wall clock", 1, 750);
    gList.addItem("AC inverter", 2, 110000);
    gList.addItem("mobile cover", 1, 250);
    gList.addItem("iron", 1, 2589);
    gList.addItem("Oranges", 345, 25);
    gList.addItem("Apples", 11, 4);
    gList.purchaseItem(2);
    gList.purchaseItem(3);
```

```
    gList.printGroceryList();
```

```
<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..LED bulb..........117.5    3      352.5
2..AC inverter.......110000   2      220000
3..iron..............2589      1      2589
4..Oranges...........25        345    8625
5..Apples............4         11     44
Total Amount _____________________________ 231610.50
```

```
    GroceryList gl;
    gl.addItem("Fans", 13, 2200);
    gl.addItem("Pen", 24, 50);
    gl.addItem("Glass", 3, 59);
    gl.purchaseItem(3);

    gList.mergeLists(gl);
```

```
    gList.printGroceryList();
```

```
<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
1..LED bulb..........117.5    3      352.5
2..AC inverter.......110000   2      220000
3..iron..............2589      1      2589
4..Oranges...........25        345    8625
5..Apples............4         11     44
6..Fans..............2200      13     28600
7..Pen...............50        24     1200
Total Amount _____________________________ 261410.50
```

```
    gl.printGroceryList();
```

```
<<<<<<<<<<<<<<<<< Grocery List >>>>>>>>>>>>>>>>>
#  Name                 Price    Count  Total
================================================
```

```
                                                1..Fans.................2200      13      28600
                                                2..Pen..................50        24      1200
                                                Total Amount                                29800.00
    gList.printPuchasedList();                  <<<<<<<<<<<<<<<<< Purchased List >>>>>>>>>>>>>>>>>
                                                #  Name                    Price     Count   Total
                                                ================================================
                                                1..wall clock..........750       1       750
                                                2..mobile cover........250       1       250
                                                Total Amount                                1000.00
    return 0;
}
```

"The hardest job kids face today is learning good manners without seeing any."

-- Fred Astaire --

In a speech Arnold Schwarzenegger said that I became very friendly with Muhammad Ali in the 70s. "Muhammad Ali worked his butt off. I saw it first-hand. I remember there was a sports writer in the gym, and Ali was doing sit-ups then. He asked Ali, how many sit-ups do you do? He said,

I don't start counting until it hurts."

Now think about it, he doesn't count until it hurts, until he feels the pain that is waking hard. So, you can't get around the hard work it doesn't matter who it is. As a maverick, I believe what the Tedd Turner said,

"Work like hell and advertise".

-- Arnold Schwarzenegger --