Objective:

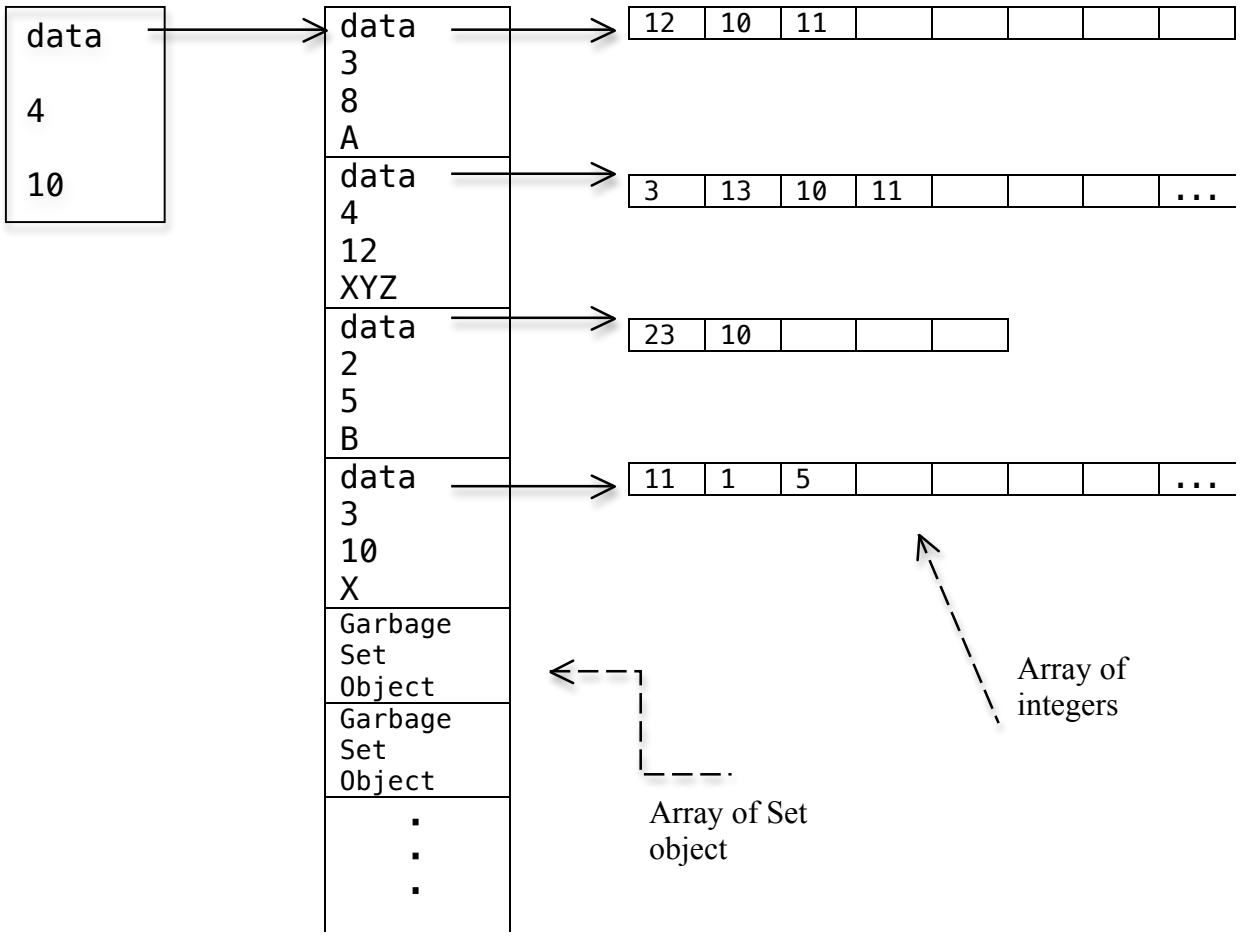
- The purpose of this quiz is to focus on the very basic fundamental concepts learned so far in previous lectures.

Following will help you to answer Question 1 and 2.
You guys should be quite familiar with struct 'Set' and struct 'SetArray'. The diagram below shows an object layout of a SetArray object named as 'sa', in which the number of sets stored by user are 4 and capacity to store sets in it is 10.

```
struct Set
{
   int * data;
   int noOfElements;
   int capacity;
   char name[30];
};
```

```
struct SetArray
{
   Set * data;
   int noOfSets;
   int capacity;
};
```

```
SetArray sa;
```



Question No. 1: (1.0)

What syntax should we write? If we have to access the second element from set of integers stored in 4th location of SetArray 'sa' object as shown in object layout above.

Question No. 2: (2.0)

Write the following function, which receives SetArray object, and deallocate all the data/array pointed by data pointer of SetArray object.
Remember: To deallocate the array of integers captured by each set object.

```
void freeSetArray ( SetArray & sa );
```

Question No. 3: (1.0)

Why C++ implicitly provides the mechanism of "this" pointer?

Because functions have single copy and the called function need to know, which object invoked him so that the called function may accordingly access the attributes of the calling object from memory.

Question No. 4: (3.0)

How is it possible to block the copying of objects of a particular class? For example for the class 'Set', user should get syntax error for doing following.

```
Set s1;
Set s2 = s1;  //must produce syntax error
```

Define copy constructor private ☺

Question No. 5: (1.0)

What enables a group of objects to belong to the same class

A. All of their attributes have identical values across all objects
B. A few selected attributes have identical values across all objects
C. All of them possess the same attributes
D. (B) and (C)

Question No. 6: (13.0)

You are to design an application, which will store sayings/quotations of different renowned philosophers/writers etc.

This application should support the following operations.

- It allows the user to add quotes along with author name (if author is not know then name it as anonymous).
- Let the user search the quotation on the basis of different keywords.

For the above application you will design an ADT 'QuotationList' with the members listed below:

Data Members:

- char * * quote; /* pointer to an array of char pointers whose each location points to a quotation (array of characters) */
- char * * author; /* pointer to an array of char pointers whose each location points to an author name(array of characters) So the quotation at quote[i] has an author at author[i] */
- int numOfQuotations; // number of quotations in quote array.
- int capacity; // size of array pointed by quote and author ***which should // be resizable***

Supported Operations:

1. `Quotation (int = 10);` (04)
   Receives the initial capacity of quote array. Default capacity is 10.

2. `void addQuotation(char * quot, char * auth="anonymous");` (04)
   Add the received quotation and author name to the quotation and author array respectively.

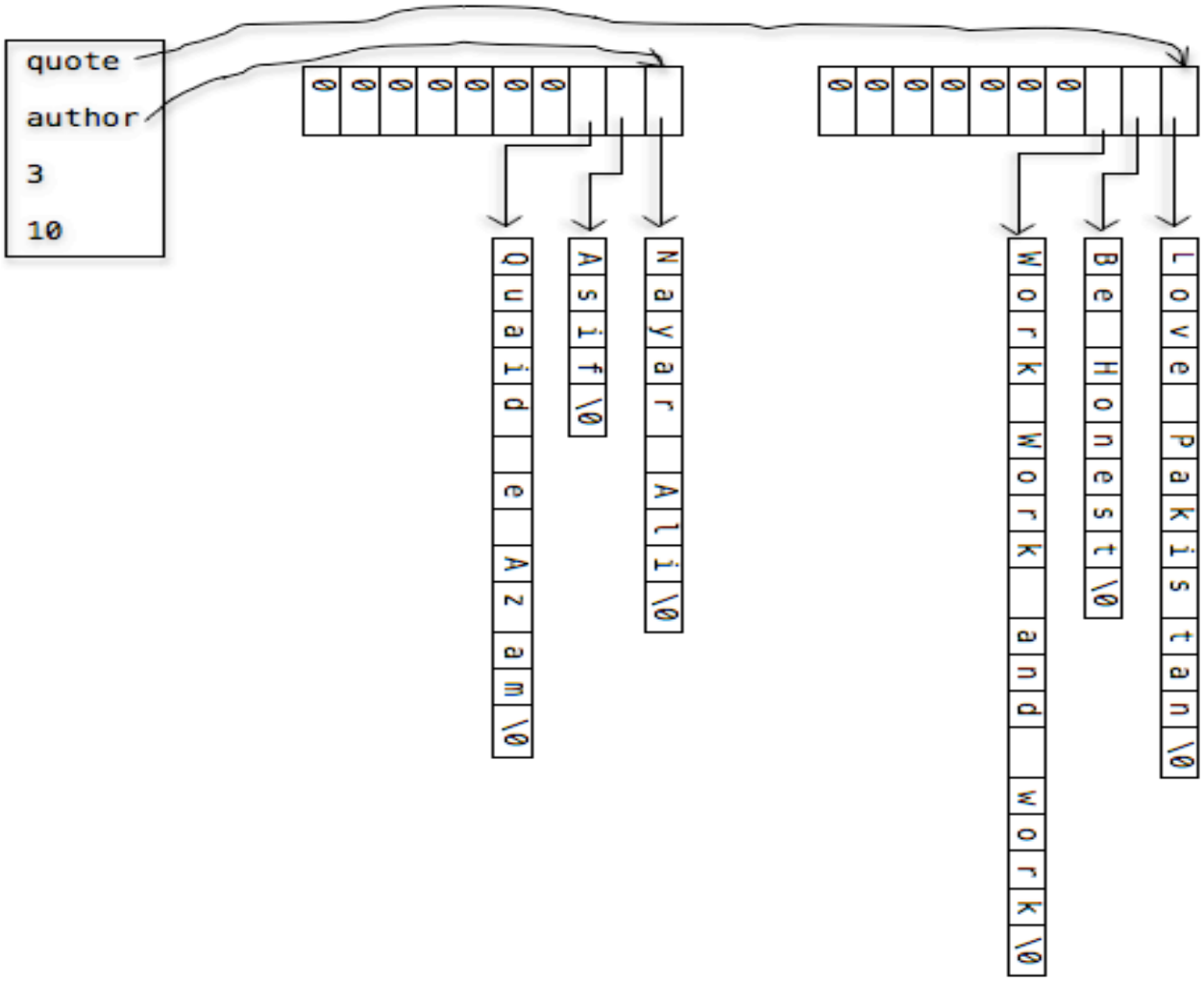
3. `void reSize( int )` (03)
   While adding quotations, if the capacity = number of quotation then addQuotation function calls this function itself, which resize the quotation and author array without loosing previous data. The new size must be double than the old size. You have to make this function private.

4. `~Quotation ();` (02)
   You know what to do.

Sample Run:

```
QuotationList ql;
ql.addQuotation("Love Pakistan","Nayar Ali");
ql.addQuotation("Be Honest","Asif");
ql.addQuotation("Work Work and Work","Quad e Azam");

//A sample object-layout/diagram is shown below, where three quotations are
//stored in QuotationList object according to the above coding statements.
```



```
class QuotationList
{
public:
    char ** quote;
    char ** author;
    int noOfQuotations;
    int capacity;
    void reSize(int nc)
    {
        char ** qt = new char* [nc];
        char ** at = new char* [nc];
        for ( int j=0; j<nc; j++)
        {
            qt[j] =0;
            at[j]=0;
        }
        int i=0;
        while( i<nc && i<noOfQuotations)
        {
            qt[i] = quote[i];
            at[i] = author[i];
            i++;
        }
        if( nc < noOfQuotations)
            noOfQuotations=nc;
        capacity =nc;
        author = at;
        quote = qt;
    }
    bool isFull()
    {
        return noOfQuotations==capacity;
    }

public:

    QuotationList(int cap=10)
    {
        if (cap<=0)
            cap=10;
        capacity=cap;
        noOfQuotations=0;
        quote = new char* [capacity];
        author = new char* [capacity];
        for ( int i=0; i<capacity; i++)
        {
            quote[i]=0;
            author[i]=0;
        }
    }
```

```
    void addQuotation(char * quot, char * auth="anonymous")
    {
        if (isFull())
            reSize(capacity*2);

        int qLen = strlen(quot);
        int aLen = strlen(auth);

        quote[noOfQuotations] = new char [qLen+1];
        author[noOfQuotations] = new char [aLen+1];
        strncpy(quote[noOfQuotations], quot, qLen);
        strncpy(author[noOfQuotations], auth, aLen);
        noOfQuotations++;
    }
    ~QuotationList()
    {
        if ( capacity==0)
            return;
        for (int i=0; i<noOfQuotations; i++)
        {
            delete [] author[i];
            delete [] quote[i];
        }
        delete [] author;
        delete [] quote;
        author=0;
        quote=0;
        noOfQuotations=0;
        capacity=0;
    }
};
```