Lab Assignment Sheet – Odd Sem 2018
Data Structure Laboratory

Instructions:

1. All students of Data Structures laboratory should be in PROPER uniform during the lab classes
2. Use of mobile phones during Lab class is STRICTLY prohibited
3. Separate instruction will be given for the mini project which is associated with this laboratory
4. Weekly schedule is planned as given below,

16-Jul	Practise Lab	23-Jul	Practise Lab	30-Jul	Practise Lab	06-Aug	Evaluation-1	13-Aug	Practise Lab	20-Aug	Project Synopsis	27-Aug	Quiz-1
17-Jul		24-Jul		31-Jul		07-Aug		14-Aug		21-Aug		28-Aug	
18-Jul		25-Jul		01-Aug		08-Aug		15-Aug		22-Aug		29-Aug	
19-Jul		26-Jul		02-Aug	IC3 2018	09-Aug		16-Aug		23-Aug		30-Aug	
20-Jul		27-Jul		03-Aug		10-Aug		17-Aug		24-Aug		31-Aug	
21-Jul		28-Jul		04-Aug		11-Aug		18-Aug		25-Aug	Holiday	01-Sep	

03-Sep	Holiday	10-Sep	Practise Lab	17-Sep	Lab Test-1	24-Sep	Practise Lab	01-Oct	Evaluation-2	08-Oct	Project Mid Evaluation	15-Oct	Practise Lab
04-Sep	T1 Exam	11-Sep		18-Sep		25-Sep		02-Oct		09-Oct		16-Oct	
05-Sep		12-Sep		19-Sep		26-Sep		03-Oct		10-Oct		17-Oct	
06-Sep		13-Sep		20-Sep		27-Sep		04-Oct		11-Oct		18-Oct	
07-Sep		14-Sep		21-Sep		28-Sep		05-Oct		12-Oct		19-Oct	
08-Sep		15-Sep		22-Sep		29-Sep		06-Oct		13-Oct		20-Oct	

22-Oct	T2 Exam	29-Oct	Quiz-2	05-Nov	MID SEM VACATIO	12-Nov	Practise Lab	19-Nov	Lab Test-2	26-Nov	Project Final Evaluation
23-Oct		30-Oct		06-Nov		13-Nov		20-Nov		27-Nov	
24-Oct		31-Oct		07-Nov		14-Nov		21-Nov		28-Nov	
25-Oct		01-Nov		08-Nov		15-Nov		22-Nov		29-Nov	
26-Oct		02-Nov		09-Nov		16-Nov		23-Nov		30-Nov	
27-Oct		03-Nov		10-Nov		17-Nov		24-Nov		01-Dec	

Week - 1 (17-Jul to 22-Jul) : Lab -Assignment (Practice Assignment)

Topics: Object oriented concepts and programming using C++

Q1 You are required to create a class **—** "Invoice". This class might be used by a departmental store to represent an invoice for an item sold at the store. An Invoice should include four data members— item number (type string), item description or name (type string), quantity of the item being purchased (type int) and price per item (type int). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named getInvoiceAmount that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as an int value. The class should be able to give the count of all the objects which are created through copy operation. Write a function that accepts two invoices for different departmental stores and

return the maximum quantity out of two invoices. Write a test program that demonstrates class Invoice's capabilities.

Q2 Create a class ―Employee that includes three pieces of information as data members—a first name (type string), a last name (type string) and a monthly salary (type int). Your class should have a constructor that initializes the three data members. Provide a set and a get function for each data member. Create two Employee objects and display each object's yearly salary. Then give each Employee a 10 percent raise and display each Employee's yearly salary again. The class should be able to give the count of all the default objects. Write a function which takes two employees and return the name of the employee with higher salary. Write a test program that demonstrates class Employee capabilities.

Q3 Create a class ―DepartmentalStore having following data members: D_Store_Name, D_Store_Address, List of Employees (consider maximum 10 employees where employees are the objects of the class type ―Employee whose details and functionalities are given in Q2), and List of invoices (consider maximum 30 invoices where invoices are the objects of the class type ―Invoice whose details and functionalities are given in Q1). Each employee is responsible to sale different items available in the store to different customers (an employee may sale items to different customers hence responsible for the generation of multiple/different invoices) and for each customer a unique invoice (contains the details and quantity of the purchased items) is supposed to be generated. Add the necessary member functions in the class DepartmentalStore and extend/update/modify (if required) the member functions/data members of the classes Invoice and Employee such that following queries can be answered, if invoked through an object of the class DepartmentalStore:

(a) Name/description of an item is inputted by the user. Output the name of the employee who sold maximum number/count of the inputted item.

(b) Based on the total amount of the sold items (in different invoices) by an employee, management of the Departmental store decided following policies: Give 10% increment to those employees who sold the items whose combined price is more than Rs. 10000 otherwise no increment is to be given to the employee. List the name of all employees who are eligible to get the increment.

(c) Give the name of the item whose total sold quantity is maximum.

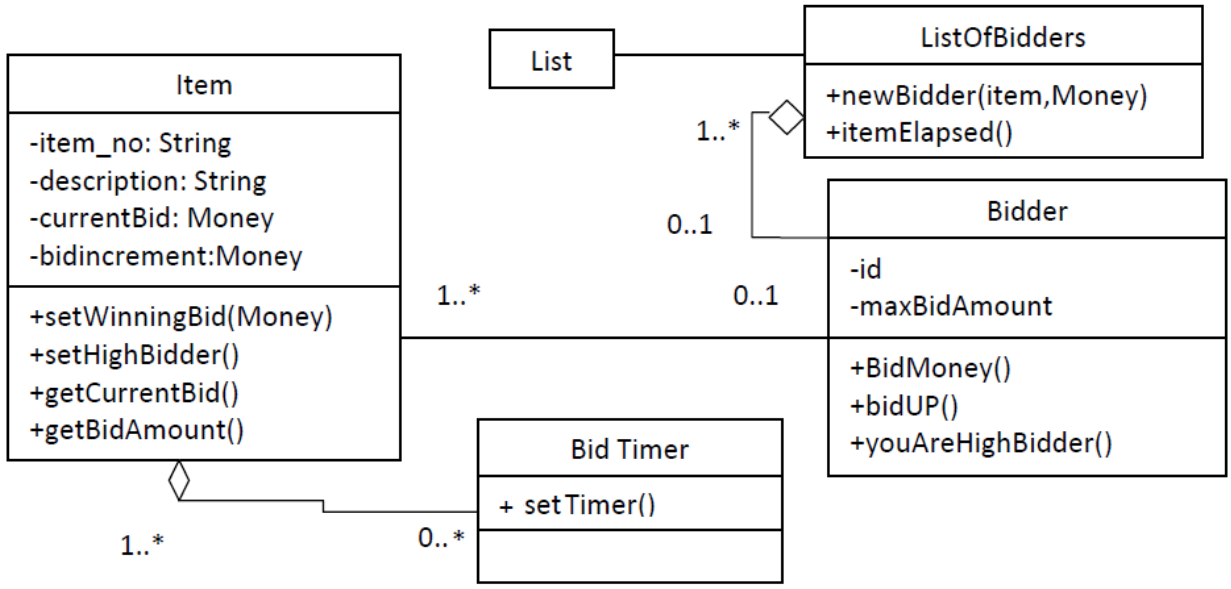
Week - 2 (23-Jul to 28-Jul) : Lab -Assignment (Practice Assignment)

Topics: Object oriented concepts and programming using C++

Q1. A class diagram is shown in figure below. Implement the classes, appropriate relationships, and respective member functions shown in the class diagram. If needed, define member functions in appropriate classes to answer following queries:

(a) Find the item and Bidder with highest bid out of a stock of 10 items

(b) List the items with current bid amount (if any).



Q2. Mobile phones are manufactured by many companies, e.g. Samsung, Sony, etc. Usually performance of these mobile phones are measured by memory (in GB), front camera (in MP), rear camera (in MP), and battery backup (in Hr) and accordingly price of the mobile phones varies. To operate mobile phones, users need sim-card which is to be issued by service provider, e.g. Airtel, Vodafone, etc. Besides the storage capacity, each sim is uniquely identified by the Mobile Number. When a sim is placed into a mobile, then only a mobile is functional otherwise not. Users need to purchase mobile as well as sim to make calls, use internet, etc. In each such phone, user can store contact details of his/her friends in the mobile storage or sim storage. To make a call, user is required to search the mobile number of his/her friend from the contact list (either of phone or of sim or of both). Whenever a call is made from one user (say X) to another user (say Y), it is required to display the message ―Airtel/Vodafone/etc. user with mobile no. XXXXXX is calling on the mobile screen of Y. Record of this call is to be stored into both users mobile phone: for X, it is outgoing and for Y, it is incoming. Further, a user may have maximum 5 bank accounts, where he/she can perform the transactions (either deposit or withdrawal). Withdrawal is based on the available balance in an account of the user. Based on the available balance in all accounts of a user, he/she decides whether a specific mobile can be purchased or not. Draw the class diagram and implement it in C++ and answer following:

(a) Who are the users capable enough to purchase a specific mobile phone model?

(b) Name the user who has made maximum call to a user (say Y, here Y can be called by many user).

Q3. It is required to create a book information system, where record of books is maintained. Any book can be categorized by the title of book, author (s) of the book, price of the book, publication year of the book, and pages. A page of a book contains the page no, texts, Figure. No, and Table No. A page may or may not have any text/figures/tables (i.e. a page may have only texts, or only figures, or only tables, or figures and tables, or figures, tables, and texts, or texts and tables, etc.). There can be maximum 3 figures in a page (if there are no texts or tables). Similarly there can be maximum 3 tables in a page (if there are no texts or figures), maximum 3 tables & figures (if there are no texts). Further, any book will have minimum 5 pages and maximum 100 pages. Every book will have an issue card where issue details (issue date, expected return date, student id, and return date) can be entered while issue or return. There can be maximum 30 entries possible in this card (i.e. total 30 records of issue/return can be stored in the book). Maximum 5 books can be issued to a student (categorized by student id, name, branch), if issued book count at any stage for a student is 5 or overdue (i.e. Exceeded the expected return date) then warning will be issued to the student and fine of the overdue books are to be imposed @2 Rs. per day. Create necessary classes for the above scenario and draw the class diagram. Write a test program that demonstrates above scenario and answer following queries:

(a) Identify which book having maximum number of figures has been issued maximum time, and

(b) Identify the name of student on which maximum fine is imposed on current day.

Q4. A grocery shop stores a variety of products which are identified through product_id, prize, name, manufacturing date. A product can be consumable & non consumable. A consumable product in addition is also having the expiry date. A grocery shop is having a no. of customers identified by the name & address. The customers can be members & non members. The members are having unique membership id while the non members are having a unique 4 digit mobile no. Grocery shop gives a discount of 20% on the total bill to the members & 2% & 4% on consumable & non consumable goods to the non members. Use the concept of inheritance & overloading to implement the above scenario. Total discount availed at the end of a day is required to be generated assuming 10 customers shopping for at max 5 products. Display the customer detail that has availed the maximum discount.

Q5. There are many publication houses in India. Publication of books written by one or more than one authors is the main activity of a publication house. A book will have different number of pages. Each page will have page number. A page may contain texts (Type 1) or texts and tables (Type 2) or texts, tables and figures (Type 3). A book will have one page of table of contents, one page of list of tables (in which page number of table is mentioned), and one page of list of figures (in which page number of figure is mentioned). Printing costs of normal page (only text) is Rs. 2, whereas for pages having texts and table is Rs. 3, and a page having texts, tables,

and figures is Rs 5. A publication house can publish many books in a month and fix price of a book which is 1.5 times of the amount needed (decided by number of pages) for publishing a book. Before publishing a book the publication house will see that whether it has sufficient fund to publish it or not. If sufficient fund is there then it will print it. Further, PH will track the record of the sold books; if it is sold the profit is added to the balance sheet of the publication house. Now consider a publisher having initial balance of Rs 1000. It published 10 books having different pages (user inputted). Say three books are being sold. Now he needs to publish a book having different types of pages. Your program will analyze whether the publisher can publish that book or not. Create necessary classes for the above scenario, draw the class diagram, and implement it to answer following queries:

a) Find total number of books published, sold. Also display the Number of pages of each published and sold book.

b) Find the revenue generated after selling the books and calculate the ratio of Revenue generate / total number of pages.

Week - 3 (30-Jul to 04-Aug) : Lab -Assignment (Practice Assignment)

Topics: Object oriented concepts and programming using C++

Q1. Four classes, C1, C2, C3, and C4 have been given in following figures along with main(). Get the output of the C++ program without running the code.

```
class C1
{
    int x;

    public:
      C1(){ x = 50; cout<<"\n Print 1"; }
      C1(int temp){ x = temp; cout<<"\n Print 2"; }
      void f1(){ cout<<"\nEnter X"; cin>>x; }
      void f2(int temp){ x = temp; }
      void f3(){ cout<<"\nX in C1 is = "<<x; }
      ~C1(){ cout<<"\n Print 3"; }
};

class C2
{
   int y;

   public:
      C2(){ y = 60; cout<<"\n Print 4"; }
      C2(int temp){ y = temp; cout<<"\n Print 5"; }
      void f4(){ cout<<"\nEnter Y"; cin>>y; }
      void f5(int temp){ y = temp; }
      void f6(){ cout<<"\nY in C2 is = "<<y; }
      ~C2(){ cout<<"\n Print 6"; }
};
```

```
class C3
{
   C1 *objC1type[2];
   C2  objC2type[2];
   public:
      C3(){ cout<<"\n Print 7";}
      C3(int temp){ cout<<"\n Print 8"; }
      void f7(int temp){ objC1type[0] = new C1; objC1type[1] = new C1(temp); }
      void f8(){ C1 *temp1 = new C1(20); C2 temp2; temp1 -> f3(); temp2.f6(); }
      void f9(C1 *temp1)
      {
            C1 *temp2, *temp3, temp4; temp2 = temp1; temp2->f2(30); temp3 = new C1;
            temp3->f2(40); temp4.f2(70);
            temp1 -> f3(); temp2 -> f3(); temp3 -> f3(); temp4.f3();
      }
      ~C3(){ cout<<"\n Print 9"; }
};
class C4
{
   C3 *obj1, obj2;
   public:
      C4(){ obj1 = new C3; cout<<"\n Print 10"; }
      void f10(){ C3 *temp1, temp2; temp1 = new C3(10); }
      ~C4(){ cout<<"\n Print 11"; }
};
```

```
int main()
{ C1 *objC1 = new C1; objC1 -> f3(); C3 objC3; C4 objC4; objC3.f7(5); objC3.f8(); objC3.f9(objC1); objC4.f10(); objC1 -> f3();  return 0; }
```

Q2. A C++ program along with main() is presented in following figures. One by one uncomment each statement in main() and analyze the output of the uncommented statement without running the code.

```
class abc
{
        int x, y;

        public:

        abc(){}
        abc(int a, int b){ x = a; y = b; }
        abc(const abc &o1){ x = o1.x; y = o1.y; }
        abc operator + (abc o1) { abc temp; temp.x = x + o1.x; temp.y = y + o1.y; cout<<"\nAddition"; return temp; }
        abc operator - (abc o1) { abc temp; temp.x = x - o1.x; temp.y = y - o1.y; cout<<"\nSubtraction"; return temp; }
        abc operator * (abc o1) { abc temp; temp.x = x * o1.x; temp.y = y * o1.y; cout<<"\nMultiplication"; return temp; }
        abc operator / (abc o1) { abc temp; temp.x = x / o1.x; temp.y = y / o1.y; cout<<"\nDivision"; return temp; }
        abc operator ++() { abc obj; obj.x = ++x; obj.y = ++y; cout<<"\nPre-Increment"; return obj; }
         abc operator ++(int temp) { abc obj; obj.x = x; obj.y = y; x++; y++; cout<<"\nPost-Increment"; return obj; }
        void display() { cout<<"\n\nX = "<<x<<" and Y = "<<y; }
};
int main()
{       abc o1(10, 20), o2(20, 40), o3;
        //o3 = o1++; o3.display();
        //o3 = ++o1; o3.display();
        //o3 = ++o1 + o1++; o3.display();
        //o3 = ++o1 + ++o2; o3.display();
        //o2 += o1; o2.display();
        //o3 = o1++ + o1++; o3.display();
        //o3 = o1++ + ++o2; o3.display();
         //o3 = ++o1 + (o1++); o3.display();
        //o3 = ++o1 + o2++ * ++o2; o3.display();
        //o3 = ++o1 - o1++ * ++o2 / o1++; o3.display();
        return 0; }
```

Q3. Three classes and their relationships are given in following figures. Analyze the given program and extend the C++ code to answer the queries (Query 1 and Query 2) shown as comments in main().

```
class subject
{
        int sub_Id; char sub_Name[20];
        int sub_SemesterOffered; int sub_BranchOfferedBy;
        public:
                subject()
                {
                        cout<<"enter subject details\n";
                        cin>>sub_Id; cin>>sub_Name;
                        cin>>sub_SemesterOffered;
                        cin>>sub_BranchOfferedBy;
                }
                char * retSubName();
                int retSub(char *tempName);
};

int subject::retSub(char *tempName)
{
        if(!strcmp(sub_Name, tempName)) return 1;
        else return 0;
}
char * subject::retSubName() { return sub_Name; }
```

```
class faculty
{
    int fac_Id; char fac_Name[20];
    char fac_Branch[10]; subject *teachesSub[3];
    int subTeachesCount;
    public:
        faculty()
        {
            cout<<"enter faculty id name branch\n";
            cin>>fac_Id; cin>>fac_Name; cin>>fac_Branch;
            subTeachesCount = -1;
        }
        void subjectTeaches(subject *temp);
};
void faculty::subjectTeaches(subject *temp)
{ teachesSub[++subTeachesCount] = temp; }
```

```
class student
{
        int std_Id, std_Semester, subRegCount;
        char std_Name[20], std_Branch[10];
        subject *registeredSub[5];

        public:
                student()
                {       cout<<"enter student details\n";
                        cin>>std_Id;
                        cin>>std_Name;
                        cin>>std_Semester;
                        cin>>std_Branch;
                        subRegCount = -1;
                }
                void subjectRegistration(subject *temp);
                int retRegSub(char *tempName);
                char * retStdName();
};
void student::subjectRegistration(subject *temp)
{
        registeredSub[++subRegCount] = temp;
}
int student::retRegSub(char *tempName)
{
        for(int t = 0; t < subRegCount; t++)
        {
                if(!strcmp(registeredSub[t]->retSubName(), tempName))
                        return 1;
        }
        return 0;
}
char * student :: retStdName()
{
        return std_Name;
}
```

```
int main()
{
        subject *sub[20]; student *std[1000]; faculty *fac[10]; int i, j, k, count; char sName[20];
        for(i = 0; i < subCount; i++) { sub[i] = new subject(); }
        for(i = 0; i < stdCount; i++) { std[i] = new student(); }
        for(i = 0; i < facCount; i++) { fac[i] = new faculty(); }
        for(i = 0; i < stdCount; i++)
        {
                cout<<"\nEnter subjects count registered by"<<i<<"th student";  cin>>count;
                for(j = 0; j < count; j++)
                {
                        cout<<"\nEnter name of "<<j<<"th subject to be registered";  cin>>sName;
                        for(k = 0; k < subCount; k++)
                        { if(sub[k] -> retSub(sName)) { std[i] -> subjectRegistration(sub[k]);  break; } }
                }
        }
        for(i = 0; i < facCount; i++)
        {
                cout<<"\nEnter subjects count taught by"<<i<<"th faculty";   cin>>count;
                for(j = 0; j < count; j++)
                {
                        cout<<"\nEnter name of "<<j<<"th subject to be taught";  cin>>sName;
                        for(k = 0; k < subCount; k++)
                        {  if(sub[k] -> retSub(sName)) {  fac[i] -> subjectTeaches(sub[k]);  break;  }  }
                }
        }
        //QUERY 1: Name of students who registered in subject OOPS.
        //QUERY 2: List all the students taught by faculty "Raj".
        return 0;
}
```

Q4. A C++ program along with main() is presented in following figure. Analyze the output of the program without running the code.

```
class a
{
        protected: int x;

        public:
                a() { x = 10; cout<<"\n1"; }   a(int t) { x = t; cout<<"\n2"; }   ~a() { cout<<"\nA is deleted"; }
};
class b : public a
{
        protected: int y;

        public:
                b() { y = 20; cout<<"\n3"; }    b(int t1, int t2):a(t1) { y = t2; cout<<"\n4"; }  ~b() { cout<<"\nB is deleted"; }
};

class c : public a
{
        protected: int z;

        public:
                c() { z = 30; cout<<"\n5"; }   c(int t1, int t2):a(t1) { z = t2; cout<<"\n6"; }  ~c() { cout<<"\nC is deleted"; }
};
class d : public b, public c
{
        int m;
        public:
                d() { m = 40; cout<<"\n7"; }
                d(int t1, int t2, int t3, int t4, int t5): b(t1,t2), c(t3,t4) { m = t5; cout<<"\n8"; }
                void f1() { cout<<"\n\n"<<b::x<<"\t"<<c::x<<"\t"<<y<<"\t"<<z<<"\t"<<m; }
                ~d() { cout<<"\nD is deleted"; }
};
int main()
{ d o2, o3(1, 2, 3, 4, 5); d *o1=new d(1, 2, 3, 4, 5);  cout<<"\nSize of O1="<<sizeof(o1); o1->f1(); delete o1; cout<<"\nSize of O2="<<sizeof(o2);
return 0; }
```

Q5. A C++ program along with main() is presented in following figure. Analyze the output of the program without running the code.

```
class a
{
        public:
                int x;
                a() { x = 10; };
                a(int t) { x = t; }
};
class b : public a
{
        public:
                int y;
                b() { y = 30; x = 40; }
                b(int t) : a(t+10) { y = t; }
};


int main()
{
        a *o1 = new a, o2(20); b o3, *o4 = new b(50); o4 = (b *)o1; cout<<o4->x; return 0;
}
```

Week - 4 (06-Aug to 11-Aug) : Lab Evaluation – 1 (10 Marks)

Syllabus: Object oriented concepts in C++ (except templates)

Week - 5 & 6 (13-Aug to 25-Aug) : Practice Lab

Q1. Write templates for the functions minimum and maximum. Minimum function should accept two arguments and return the value of the arguments that is the lesser among the two. Maximum function should accept two arguments and return the value of the arguments that is the greater among the two values. Design a simple driver program that demonstrates the templates with various data types.

Q2. Build selection and insertion sorts as templates. Test these with four different data types out of which one will be string and other will be an object of a class of your choice. Define your own functions for comparison if required.

Q3. Build a generic class array. Use it to define two arrays, one of integers and the other of real numbers. Is it possible to define these arrays to be of different sizes? If yes, then define the array of integers to have a size 10 and the one of real's to have a size of 20. The operations that can be performed on the arrays are the sum and multiplication. The former add up all the elements whereas the latter multiplies the elements. Each operation prints the final result.

Q4. Vector is one of the often used containers (STL) in C++ and briefly defined as follows: Vectors are used when dynamic size arrays are needed. Like arrays, vector elements are placed in contiguous storage location so they can be accessed and traversed using iterators. To traverse the vector we need the position of the first and last element in the vector. Considering the various functionalities of Vector, create your own template class MyVector with data members and member functions such that all the functionalities of Vector can also be performed by MyVector.

Q5. Set is one of the often used containers (STL) in C++ and briefly defined as follows: Sets are containers which store only unique values and permit easy look ups. The values in the sets are stored in some specific order (like ascending or descending). Elements can only be inserted or deleted, but cannot be modified. We can access and traverse set elements using iterators just like vectors. Some of the important functionalities of Set are as follows:

begin (): Returns an iterator to the first element of the set. Operational time is O (1).

clear (): Deletes all the elements in the set and the set will be empty. Operational time is O (N), where N is the size of the set.

count (): Returns 1 or 0 if the element is in the set or not respectively. Operational time is O (log N) where N is the size of the set.

empty (): Returns true if the set is empty and false if the set has at least one element. Operational time is O (1).

end (): Returns an iterator pointing to a position which is next to the last element. Operational time is O (1).

erase (): Deletes a particular element or a range of elements from the set. Operational time is O(N) where N is the number of element deleted.

find (): Searches for a particular element and returns the iterator pointing to the element if the element is found otherwise it will return the iterator returned by end().Operational time is O (log N) where N is the size of the set.
insert (): insert a new element. Operational time is O (log N) where N is the size of the set.
size (): Returns the size of the set or the number of elements in the set. Operational time is O (1).
Observing the functionalities of Set, create your own template class MySet and facilitate it with data members and member functions so that all the above mentioned functionalities of Set can also be performed by MySet.
Q6. List is one of the often used containers (STL) in C++ and briefly defined as follows: List is a sequence container which takes constant time in inserting and removing elements. List in STL is implemented as Doubly Link List. The elements from List cannot be directly accessed. For example to access element of a particular position, you have to iterate from a known position to that particular position. Considering the various functionalities of List, create your own template class MyList with data members and member functions such that all the functionalities of List can also be performed by MyList.
Q7. Create a template class **Sort** capable to sort an array (int, float, or char type) using following sorting techniques: Selection sort Bubble sort Merge sort Quick sort (recursive) Radix sort Bucket sort
i.e. if an object O defined as Sort <int> O, calls a member function QuickSort as O.QuickSort(A), where A is an integer array will sort the array A.
Q8. An array **ARR** of n elements (where elements might not be unique) is given. It is desired to construct an array **ARRNEW** from ARR where elements of ARRNEW are the unique elements of ARR, i.e. if there are N elements in ARR out of which K ($K \leq N$) elements are unique then size of the array ARRNEW is K and will contain all the unique K elements. Propose and implement an efficient scheme to create the array ARRNEW.
Q9. You have been given two sorted lists of size M and N. It is desired to find the Kth smallest element out of M+N elements of both lists. Propose and implement an efficient algorithm to accomplish the task. Further, propose and implement an efficient algorithm to accomplish the task considering that elements in both lists are unsorted.
Q10. You have been given a sorted array **ARR** (of size M, where M is very large) of two elements, 0 and 1. It is desired to compute the count of 0s in the array ARR. Propose and implement an efficient algorithm to accomplish the task.

Week - 7 (27-Aug to 01-Sep) : Quiz-1 (10 Marks)

Syllabus: Templates, STL and sorting

Week - 8 (03-Sep to 10-Sep) : T1 Examination

Week - 9 (11-Sep to 17-Sep) : Practice Lab

Q1. Practice questions - Q1, Q2 and Q3 from T1 examination paper

Week - 10 (18-Sep to 23-Sep) : Lab Test-1

Syllabus: OOP, Class diagram, Polymorphism, templates, STL, sorting and searching

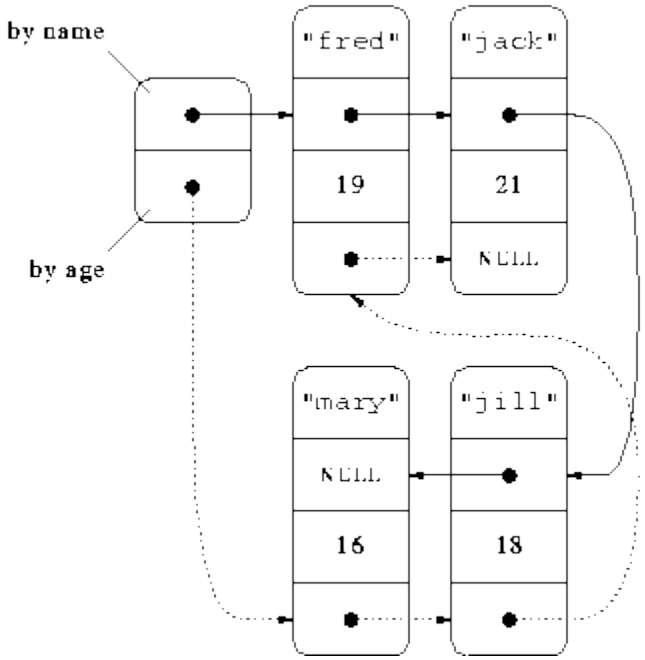
Week - 11 (24-Sep to 29-Sep) : Practice Lab

Q1. The standard use of multi-linked lists is to organize a collection of elements in two different ways. For example, suppose the elements in node include the name of a person and his/her age. e.g.

(FRED,19) (MARY,16) (JACK,21) (JILL,18)

There is need to order these elements alphabetically and also to order them by age. There will be need for two pointers - NEXT-alphabetically, NEXT-age - and the list header would have two pointers, one based on name, the other on age [as shown in figure below].
Implement the following operations,

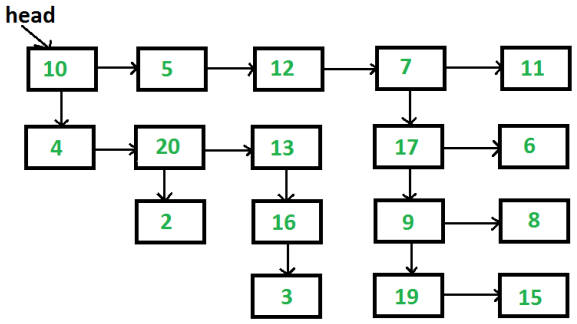
a) Insertion of new nodes in the multilinked list such that the list is sorted after insertion
b) Deletion of a node from the multilinked list based on the user inputted age/name
c) Display the multilinked list



Q2. Given a linked list where in addition to the next pointer, each node has a child pointer, which may or may not point to a separate list. These child lists may have one or more children of their own, and so on, to produce a multilevel data structure, as shown in below figure. You are given the head of the first level of the list. Flatten the list so that all the nodes appear in a single-level linked list. Write the program to flatten the list in way that all nodes at first level should come first, then nodes of second level, and so on. Each node of linked list is defined as below,

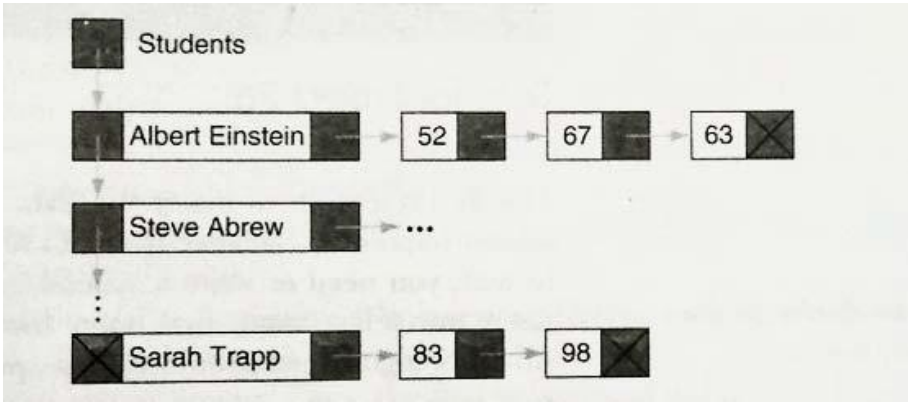
```
class node
{
private:
        int data;
        node *next;
```

```
        node *child;
public:

/* All the required get and set functions. Constructors are also defined*/

}*head;
```



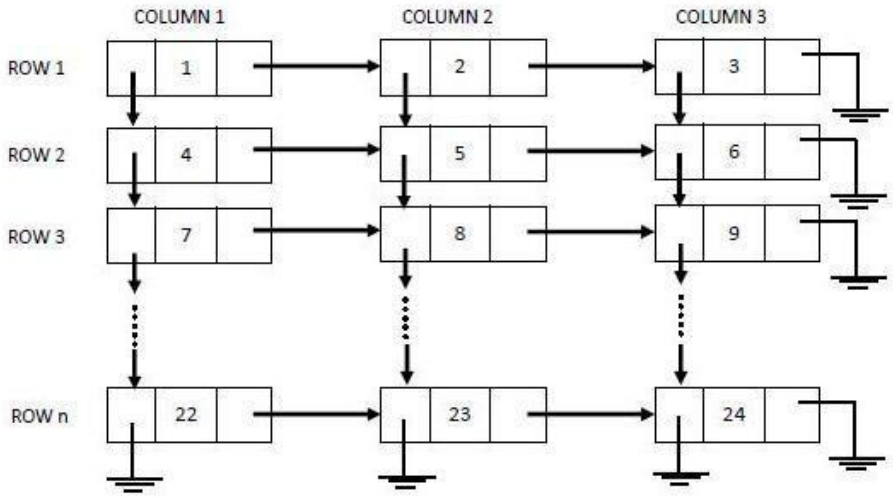
The above list should be converted to 10->5->12->7->11->4->20->13->17->6->2->16->9->8->3->19->15

Q3. Write a program to read a list of students from a file and create a linked list. Each entry in the linked list should have the student's name, a pointer to the next pointer, and a pointer to a linked list of scores. There may be up to 4 scores for each student. The program should initialize the student list by reading the students' names from the file and creating null score lists.

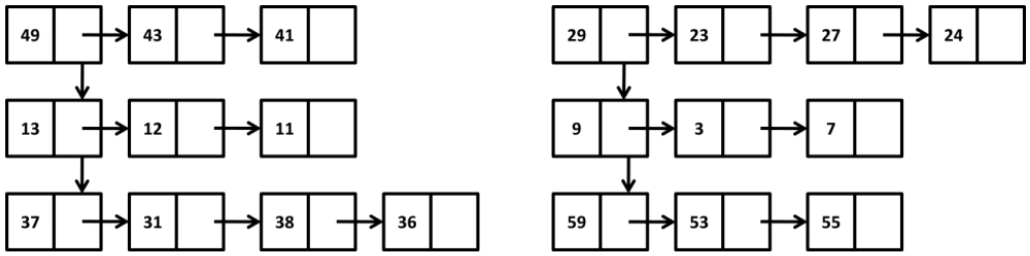


It should then loop through the list, prompting the user to enter the scores for each student. The scores' prompt should include the name of the student. After all scores have been entered, the program should print the scores for each student along with the scoretotal and avg. The average should include only those scores present.

Q4. Write a program to create a data structure as indicated in following figure. Further display the contents of this structure (a) Row wise (b) Column Wise



Q5. Create and display two multi-lists as shown below



Multi-list 1 Multi-list 2

a) Write a function to sort the elements in the multi-lists in ascending order. Display the sorted multi-lists individually.

b) Write a function to merge the sorted multi-lists obtained in Question A. Display the multi-list after merging. Note that the elements in the merged multi-list should be in ascending order at each level.

Q6. You have been given a list of characters as {A, B, C, D, E, F, G, H, and I). Write a program to insert these elements into a complete binary tree. Also, write the code to perform all the functionalities on the constructed binary tree,

a. To find a node with specified value
b. To compute height of a given node
c. To compute the depth of a node
d. To check whether the given tree is complete binary tree or not

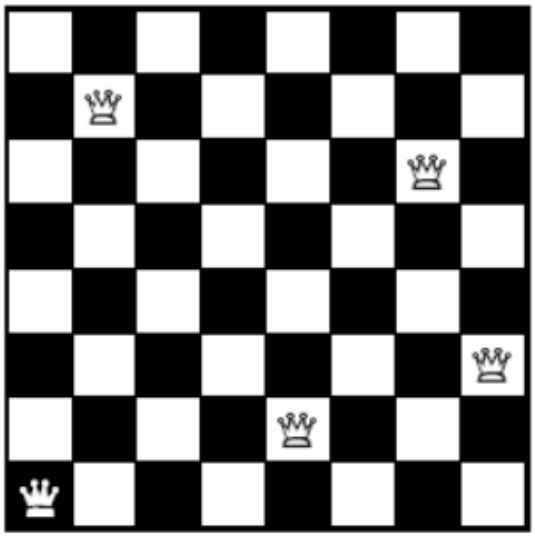
Q7. Write a program to construct a binary tree from its level order and in-order traversal.

Week - 12 (01-Oct to 06-Oct) : Practice Lab

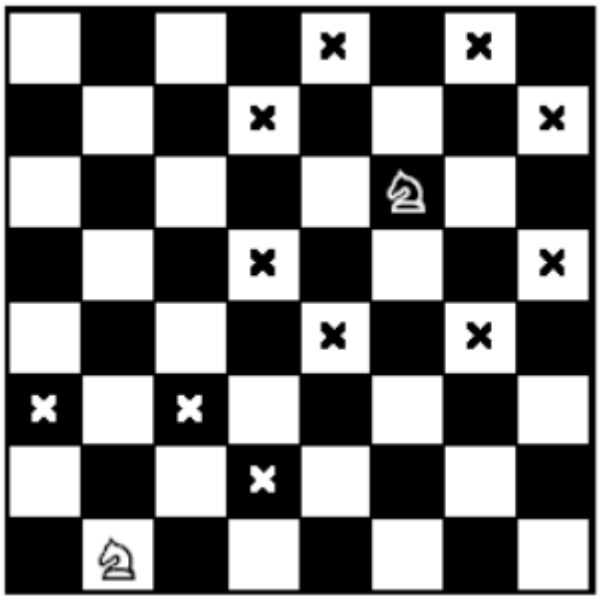
Evaluation-2 will be conducted based on the questions practiced in Week-11 Lab class

Q1. The most powerful piece in the game of chess is the queen, which can move any number of squares in any direction, horizontally, vertically, or diagonally. It is possible to place eight queens on an 8x8 chessboard so that none of them attacks any of the others, as shown in the following diagram. Write a program that solves the more general problem of whether it is possible to place N queens on an NxN chessboard so that none of them can move to a square

occupied by any of the others in a single turn. Your program should either display a solution if it finds one or report that no solutions exist.

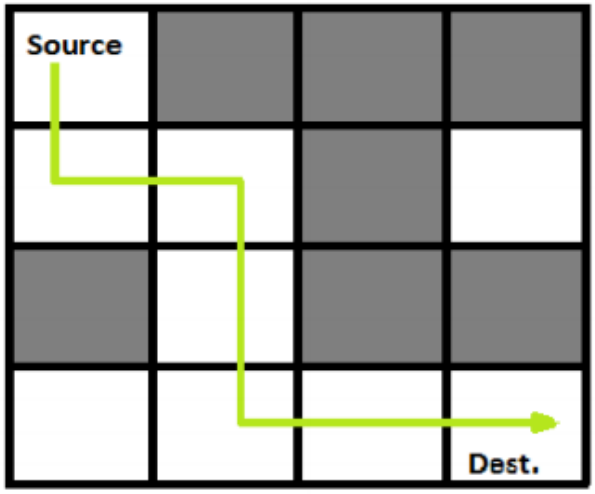


Q2. In chess, a knight moves in an L-shaped pattern: two squares in one direction horizontally or vertically, and then one square at right angles to that motion. For example, the knight in the upper right side of the following diagram can move to any of the eight squares marked with a black cross:



It turns out that a knight can visit all 64 squares on a chessboard without ever moving to the same square twice. A path for the knight that moves through all the squares without repeating a square is called a knight's tour. Write a program to find out Knight's tour.

Q3. Given a maze as an NxN matrix, a rat has to find a path from source to destination. The matrix maze[0][0] (left top corner) is the source and maze[N-1][N-1](right bottom corner) is destination. There are few cells which are blocked, means rat cannot enter into those cells. Rat can move in any direction (left, right, up and down).

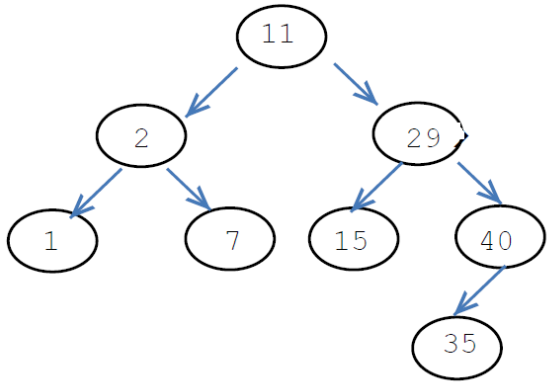


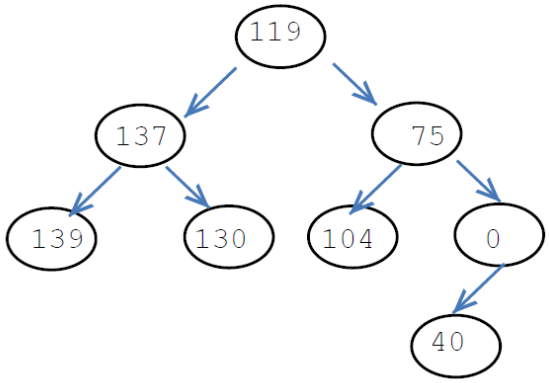
Q4. WAP to find the K^{th} smallest element in a BST.

Q5. A full K-ary tree is a tree where each node has either 0 or K children. Given an array, that contains the preorder traversal of full k-ary tree, WAP for constructing the full k-ary tree using this array. (Hint: In k-ary tree, for a node at ith position its children will be at k*i+1 to k*i+k)

Q6. Given a binary tree and inorder traversal, construct a new binary tree with additional pointers such that the inorder traversal is reflected in the new tree (Hint: Threaded binary tree).

Q7. Given a BST, transform it into greater sum tree where each node contains sum of all nodes greater than that node.





Q8. Construct a binary tree from its level order and in-order traversal.

Q9. In a binary search tree, write a function to checks if the sum of all the nodes in left sub tree of the node is equal or greater or lesser than the sum of all the nodes in right sub tree of the node.