Tutorial and Assignment Sheet – ODD 2019
15B11CI311 – Data Structures
Course Coordinator: Manish K Thakur

Team

Lecture: Deepti Singh, Manju, Manish Kumar Thakur, Sherry Garg, Vivek Kumar Singh

Tutorial: Dr. Anita Sahu, Deepti Singh, Dr. Manish K Thakur, Prantik Biswas, Sherry Garg, Prof. Vikas Saxena

Week Details

	Intro Week	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	T1 Week	Week 9	
Mon		22-Jul	29-Jul	05-Aug	12-Aug	19-Aug	26-Aug	02-Sep	09-Sep	16-Sep	23-Sep	Mon
Tue		23-Jul	30-Jul	06-Aug	13-Aug	20-Aug	27-Aug	03-Sep	10-Sep	17-Sep	24-Sep	Tue
Wed	17-Jul	24-Jul	31-Jul	07-Aug	14-Aug	21-Aug	28-Aug	04-Sep	11-Sep	18-Sep	25-Sep	Wed
Thu	18-Jul	25-Jul	01-Aug	08-Aug	15-Aug	22-Aug	29-Aug	05-Sep	12-Sep	19-Sep	26-Sep	Thu
Fri	19-Jul	26-Jul	02-Aug	09-Aug	16-Aug	23-Aug	30-Aug	06-Sep	13-Sep	20-Sep	27-Sep	Fri
Sat	20-Jul	27-Jul	03-Aug	10-Aug	17-Aug	24-Aug	31-Aug	07-Sep	14-Sep	21-Sep	28-Sep	Sat
Sun	21-Jul	28-Jul	04-Aug	11-Aug	18-Aug	25-Aug	01-Sep	08-Sep	15-Sep	22-Sep	29-Sep	Sun

COLOR CODE | HOLIDAY | EXAMS: T1/T2 | IC3

	Week 10	Week 11	Week 12	T2 Week & Mid Sem		Week 13	Week 14	Week 15	Week 16 (Doubt Week)			
Mon	30-Sep	07-Oct	14-Oct	21-Oct	28-Oct	04-Nov	11-Nov	18-Nov	25-Nov	02-Dec		Mon
Tue	01-Oct	08-Oct	15-Oct	22-Oct	29-Oct	05-Nov	12-Nov	19-Nov	26-Nov	03-Dec		Tue
Wed	02-Oct	09-Oct	16-Oct	23-Oct	30-Oct	06-Nov	13-Nov	20-Nov	27-Nov			Wed
Thu	03-Oct	10-Oct	17-Oct	24-Oct	31-Oct	07-Nov	14-Nov	21-Nov	28-Nov			Thu
Fri	04-Oct	11-Oct	18-Oct	25-Oct	01-Nov	08-Nov	15-Nov	22-Nov	29-Nov			Fri
Sat	05-Oct	12-Oct	19-Oct	26-Oct	02-Nov	09-Nov	16-Nov	23-Nov	30-Nov			Sat
Sun	06-Oct	13-Oct	20-Oct	27-Oct	03-Nov	10-Nov	17-Nov	24-Nov	01-Dec			Sun

Instructions:

(1) Evaluation Scheme:

Components	**Maximum Marks**
T1	20
T2	20
End Term Examination	35
TA	25 (Mini Project (10), Attendance (5), Tutorial Assignments (5), Online Test (5))
Total	**100**

(2) Week wise tutorial sheet will be updated on Monday of each week. A detail of various weeks in ODD 2019 for Data Structure is elaborated at Week Details.

(3) Besides the assignments, we will be posting important notices on this sheet throughout the semester.

(4) Mini-project made in associated lab of this course contributes 10 marks out of 100

(5) As per University norms, all students are required to ensure 80% attendance in Lecture and Tutorials and this component contributes 5 marks out of 100

Introduction Week (17th July to 20th July 2019)

1. Review of various operations like insert, delete, traverse, search on Linked List using OOPS
2. Review of Class, Object, Constructors and Destructors in C++
3. Static and dynamic memory allocation for arrays and linked list

Week# 1 (22nd July to 27th July 2019)

Topics: Memory Allocation, Relationships in OOPS, Class and members

Q1. Analyze the correctness and output of following programs

#include <iostream>
#include <malloc.h>
using namespace std;
int main() {
float *a;
a = (float *)malloc(sizeof(int));
a[0] = 4.5;
cout<<a[0];
return 0;
}
	#include <iostream>
	#include <malloc.h>
	using namespace std;
	int main() {
	int *a;
	a = (int *)malloc(sizeof(float));
	a[0] = 5;
	cout<<a[0];
	return 0;
	}

#include <iostream>
#include <malloc.h>
using namespace std;
int main() {
int *a, *b;
a = (int *)malloc(sizeof(int));
b = (int *)malloc(5*sizeof(int));
cout<<sizeof(a)<< sizeof(b);
return 0;
}
	#include <iostream>
	#include <malloc.h>
	using namespace std;
	int main() {
	int *a;
	a[0] = (int *)malloc(sizeof(int));
	a[0] = 5;
	cout<<a[0];
	return 0;
	}

#include <iostream>
#include <malloc.h>
using namespace std;
int main() {
int *a[5];
a[0] = (int *)malloc(sizeof(int));
a[0][0] = 5;
cout<<a[0][0];
return 0;
}
	#include <iostream>
	#include <malloc.h>
	using namespace std;
	int main() {
	struct node{int a[10];};
	struct node *n;
	n = (struct node *)malloc(sizeof(struct node));
	cout<<sizeof(n);
	return 0;
	}

#include <iostream>
#include <malloc.h>
using namespace std;
int main() {
int *a[5];
a[0] = (int *)malloc(2*sizeof(int));
a[0][1] = 5;
cout<<a[0][1];
return 0;
}
	#include <iostream>
	#include <malloc.h>
	using namespace std;
	int main() {
	int *a = (int *)malloc(5*sizeof(int));
	a[0] = 1; a[1] = 2; a[2] = 3; a[3] = 4; a[4] = 5;
	delete(a);
	cout<<a[0]<<a[1]<<a[2]<<a[3]<<a[4];
	return 0;
	}

Q2. 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.

Q3. 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.

Q4. Identify the relationship(s); analyze the sequence of constructor and destructor invocation; and obtain the output

```
class ABC
{
    int x;
    public:
     ABC() { cout<<"1"; }
     ~ABC() { cout<<"2"; }
};
```

```
class KLM
{
    int y;
    ABC O1;
    public:
     KLM() { cout<<"3"; }
     ~KLM() { cout<<"4"; }
};
```

```
class XYZ
{
    int y; KLM O2;
    ABC O3;
    public:
     XYZ() { cout<<"5"; }
     ~XYZ() { cout<<"6"; }
};
```

```
int main() { XYZ O4, *O5; KLM *O6; O5 = new XYZ; O6 = new KLM; delete(O5); return 0; }
```

Q5. Identify the relationship(s); analyze the sequence of constructor and destructor invocation; and obtain the output

```
class ABC
{
    int x;
    public:
     ABC() { cout<<"1"; }
     ~ABC() { cout<<"2"; }
};
```

```
class KLM
{
    int y;
    ABC *O1;
    public:
     KLM() { cout<<"3"; }
     ~KLM() { cout<<"4"; }
};
```

```
class XYZ
{
    int y; KLM O2;
    ABC O3;
    public:
     XYZ() { cout<<"5"; }
     ~XYZ() { cout<<"6"; }
};
```

```
int main() { XYZ O4, *O5; KLM *O6; O5 = new XYZ; O6 = new XYZ; delete(O5); return 0; }
```

Q6. Identify the relationship(s); analyze the sequence of constructor and destructor invocation; and obtain the output

```
class ABC
{
    int x;
    public:
     ABC() { cout<<"1"; }
     ~ABC() { cout<<"2"; }
};
```

```
class KLM
{
    int y;
    ABC *O1;
    public:
     KLM() { cout<<"3"; O1 = new ABC;}
     ~KLM() { cout<<"4"; }
};
```

```
class XYZ
{
    int y;
    KLM O2;
    ABC O3;
    public:
     XYZ() { cout<<"5"; }
     ~XYZ() { cout<<"6"; }
};
```

```
int main() { XYZ O4, *O5; KLM *O6; O5 = new XYZ; O6 = new KLM; delete(O5); return 0; }
```

Q7. Identify the relationship(s); analyze the sequence of constructor and destructor invocation; and obtain the output

```
class ABC
{
    int x;
    public:
     ABC() { cout<<"1"; }
     ~ABC() { cout<<"2"; }
};
```

```
class KLM
{
    int y;
    ABC *O1;
    public:
     KLM() { cout<<"3"; O1 = new ABC;}
     ~KLM() { cout<<"4"; delete(O1);}
};
```

```
class XYZ
{
    int y;
    ABC O2;
    KLM O3;
    public:
     XYZ() { cout<<"5"; }
     ~XYZ() { cout<<"6"; }
};
```

```
int main() { XYZ O4, *O5; KLM *O6; O5 = new XYZ; O6 = new XYZ; delete(O5); return 0; }
```

Week# 2 (29th July to 3rd August 2019) & Week# 3 (5th August to 10th August 2019)

Topics: Relationships in OOPS, Class Diagram, Polymorphism

Q1. Identify the relationship(s); analyze the sequence of constructor invocation; and obtain the output

```
class A
{
    int x;
    public:
        A()
        {cout<<" 1";}
        A(int k)
        {cout<<" 2";}
};
```

```
class B
{
    A o1;
    public:
        B()
        {cout<<" 3";}
        B(int k): o1(k)
        {cout<<" 4";}
};
```

```
class C
{
    B o1, *o2; A o3;
    public:
        C(): o3(5)
        {o2 = new B;
            cout<<" 5";}
        C(int k): o3(k), o1(k)
        {o2 = new B;
            cout<<" 6";}
};
```

```
class D
{
    C o1; B o2; A o3;
    public:
        D()
        {cout<<" 7";}
        D(int k): o3(k)
        {cout<<" 8";}
        D(int k, int l): o2(k), o1(l)
        {cout<<" 9";}
};
```

```
int main()
{D o1; cout<<"\n"; D o2(5); cout<<"\n"; D o3(5, 10); return 0; }
```

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. 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.

Q4. 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.

Q5. Analyze the sequence of constructor invocation; and obtain the output

class A
{
public:
A()
{cout<<" 1";
}
};
	class B : virtual public A
	{
	public:
	B()
	{
	cout<<" 2";
	}
	};
		class C : public B
		{
		public:
		C()
		{
		cout<<" 3";
		}
		};
			class D : public C
			{
			public:
			D()
			{
			cout<<" 4";
			}
			};
				class E
				{
				public:
				E()
				{
				cout<<" 5";
				}
				};

class F : public E
{
public:
F()
{cout<<" 6";
}
};
	class G : virtual public F
	{
	public:
	G()
	{ cout<<" 7";
	}
	};
		class H : public G
		{
		public:
		H()
		{ cout<<" 8";
		}
		};
			class I : public H
			{
			public:
			I()
			{ cout<<" 9";
			}
			};
				class J : public D, public I
				{
				public:
				J()
				{ cout<<" 10";
				}
				};

int main(){J O; return 0;}				

Q6. A job seeker enquires with one or more employment agencies (at max 5). The employment agencies keep a track on different types of employment such as RetailPositions, LabourerPositions and InfoTechPositions. RetailPosition class has attributes such as employer name, positions available, location and pay rate. LabourerPositions has attributes such as employer name, positions available, location, daily wage and skill set required. InfoTechPosition class has attributes such as employer name, positions available, location salary, degree and experience required. The employment agency keeps track of the jobs available in each category of employment. When a job seeker needs to search

for a job, he/she has to contact the agency and provide his personal details such as name, highest degree, experience details, skill set and expected pay rate/salary. In turn the employment agency provides the seeker with the list of jobs available in each of the category and searches of the particular suitable job for the seeker based on the type of job. The search differs in each of the category such as the RetailPositions can be searched using pay rate and in case of LabourerPositions, the search is based on the skills set. In case of InfoTechPositions, the search criterion is based on the degree and experience. The agency retrieves the list of jobs available in that category and displays it to the job seeker. Identify the classes and its relations. Also Code the following case in C++ showing all possible object-oriented concepts including inheritance and run time polymorphism.

Week# 4 (12th Aug to 18th August 2019) & Week# 5 (19th Aug to 24th August 2019)

Topics: Polymorphism and Template

Q1. Draw the class diagram from following code, identify the abstract classes, and perform following queries (you may include member functions in different classes to perform the required queries):

(a) Find out the topper (print the enrolment number) of the subject (subject ID is user inputted)
(b) Find out the list of the students who are debarred in a subject (subject ID is user inputted)
(c) Find out the topper of a semester (a semester between 1 and 8 is user inputted)

```
class subject
{
    protected:
        int subID, credit, semester, attndCutoff;
    public:
        subject() {
        int x;  x = rand()%10000+10000;  subID = x;
        x = rand()%8+1;  semester = x;  }
        int returnCredit()  { return credit; }
        void prntSubID() { cout<<subID; }
        virtual int scoreMarks() = 0;
        virtual int debarStatus(int x) = 0;
};

class core : public subject
{
    public: core() { }
};
```

```
class student
{
protected:
    int stdID, semester, *attendance, *marks;
    subject *sub[6];   float CGPA;
public:
    student()
    {
        stdID = rand()%100000 + 100000;
        semester = rand()%8 + 1;
        CGPA = 0.0;
    }
    int getSemester()  {  return semester;  }
    virtual void printDetails() = 0;
    virtual void computeCGPA() = 0;
    virtual void giveExam() = 0;
    virtual void registerSubject(subject *o) = 0;
};
```

```
class Theory : public core
{
public:
    Theory() { credit = 4;
    attndCutoff = 70;  }
    int debarStatus(int x)
    {
        int tutorialAttnd;
        tutorialAttnd = rand()%90 + 10;
        if((x+tutorialAttnd) >= attndCutoff)
            return 1;
        else
            return 0;
    }
    int scoreMarks()
    {
        int T1, T2, T3, TA, Tutorial;
        T1 = rand()%21;  T2 = rand()%21;
        T3 = rand()%36;
        TA = rand()%16;
        Tutorial = rand()%11;
        return T1+T2+T3+TA+Tutorial;
    }
};
```

```
class Lab : public core
{
public:
    Lab(){credit = 1;
    attndCutoff = 80;}
    int debarStatus(int x)
    {
        if(x >= attndCutoff)
            return 1;
        else
            return 0;
    }
    int scoreMarks()
    {
        int LT1, LT2, Assgn;
        int T, MiniPrj;
        LT1 = rand()%21;
        LT2 = rand()%21;
        MiniPrj = rand()%11;
        Assgn = rand()%51;
        T = LT1+LT2+Assgn+MiniPrj;
        return T
    }
};
```

```
class Elective : public subject
{
public:
    Elective()
    {
        credit = 3;
        attndCutoff = 70;
    }
    int debarStatus(int x)
    {
        if(x >= attndCutoff)
            return 1;
        else { return 0; }
    }
    int scoreMarks()
    {
        int T1, T2, T3, TA;
        T1 = rand()%21;
        T2 = rand()%21;
        T3 = rand()%36;
        TA = rand()%26;
        return T1+T2+T3+TA;
    }
};
```

```
class UGStudent : public student
{
public:
	UGStudent()
	{
		if(semester%2 == 0) {  marks = new int[6];   attendance = new int[6];   for(int i = 0; i < 6; i++) {  sub[i] = NULL; } }
		else	{ marks = new int[5];  attendance = new int[5];   for(int i = 0; i < 6; i++) {  sub[i] = NULL;  } }
	}
	void registerSubject(subject *o)
	{
		int j, i = 0;
		if(semester%2 == 0) { j = 6;  }
		else {  j = 5;  }
		while(sub[i] != NULL && i < j) {   i++;   }
		if(i == j) {  cout<<"\nCan not register more than "<<j<<" subjects"; }
		else {  sub[i] = o;  }
	}
	void giveExam()
	{
		int j, i;
		if(semester%2 == 0) {  j = 6;  }
		else {  j = 5;  }
		for(i = 0; i < j; i++)
		{
			attendance[i] = rand()%101;
		}
		for(i = 0; i < j; i++)
			if(sub[i] -> debarStatus(attendance[i])) {  marks[i] = sub[i] -> scoreMarks();  }
			else {   marks[i] = 0;   }
	}
	void computeCGPA()
	{
		int j, i, credit, creditTotal = 0, gradePoint = 0;
		if(semester%2 == 0) {  j = 6;  }
		else {  j = 5;  }
		for(i = 0; i < j; i++)
		{
			credit = sub[i] -> returnCredit(); creditTotal = creditTotal + credit;
			if(marks[i] >= 90) {  gradePoint = gradePoint + credit * 10;  }
			if(marks[i] >= 80 && marks[i] < 90) {  gradePoint = gradePoint + credit * 9;  }
			if(marks[i] >= 70 && marks[i] < 80) {  gradePoint = gradePoint + credit * 8;  }
			if(marks[i] >= 60 && marks[i] < 70) {  gradePoint = gradePoint + credit * 7;  }
			if(marks[i] >= 50 && marks[i] < 60) {  gradePoint = gradePoint + credit * 6;  }
			if(marks[i] >= 40 && marks[i] < 50) {  gradePoint = gradePoint + credit * 5;  }
			if(marks[i] >= 30 && marks[i] < 40) {  gradePoint = gradePoint + credit * 4;  }
			if(marks[i] < 30) {  gradePoint = gradePoint + credit * 0;  }
		}
		CGPA = gradePoint/creditTotal;
	}
	void printDetails()
	{
		cout<<"\nStudent ID: "<<stdID;
		if(semester%2 == 0)
		{
		  for(int i = 0; i < 6; i++)
		  {
			sub[i] -> prntSubID();  cout<<" Attend = "<<attendance[i]<<" Marks = "<<marks[i]<<"\n";
		  }
		}
		cout<<"\nCGPA = "<<CGPA;
	}
};
```

```
int main()
{
	subject *subj[50]; student *stdu[200]; int i, j, k, m;
	for(i = 0; i < 50; i++)
	{
		j = rand()%3;
		if(j == 0) { subj[i] = new Theory; }
		if(j == 1) { subj[i] = new Lab; }
		if(j == 2) { subj[i] = new Elective; }
	}
	for(i = 0; i < 200; i++)
	{
		stdu[i] = new UGStudent;  j = stdu[i] -> getSemester();
		if(j % 2 == 0) { for(k = 0; k < 6; k++) { m = rand()%50; stdu[i] -> registerSubject(subj[m]); } }
		else { for(k = 0; k < 5; k++) { m = rand()%50; stdu[i] -> registerSubject(subj[m]); } }
	}
	for(i = 0; i < 200; i++) { stdu[i] -> giveExam(); stdu[i] -> computeCGPA(); }
	j = rand()%200;
	stdu[j] -> printDetails();
	return 0;
}
```

Q2. Define a template class for complex numbers which takes different types of data except character and string type. Develop the addition, subtraction, and multiplication operation on complex numbers in it. Write a test function to check the various operations on complex numbers using different types.

Q3. Using a stack STL, write a program to convert the expression w1*f1*1/2+w2*f2*1/2+w3*f3 into a postfix form.

Q4. Using a map STL, write a program to store the details of person such as Aadhar number, name, DOB, place of birth, address and city. Using STL function, retrieve and display the details of persons whose Aadhar number has two odd numbers followed by an even number (e.g., 13456789, 24633871).

Q5. 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.

Q6. 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.

Q7. 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.

Q8. 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.

Q9. 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

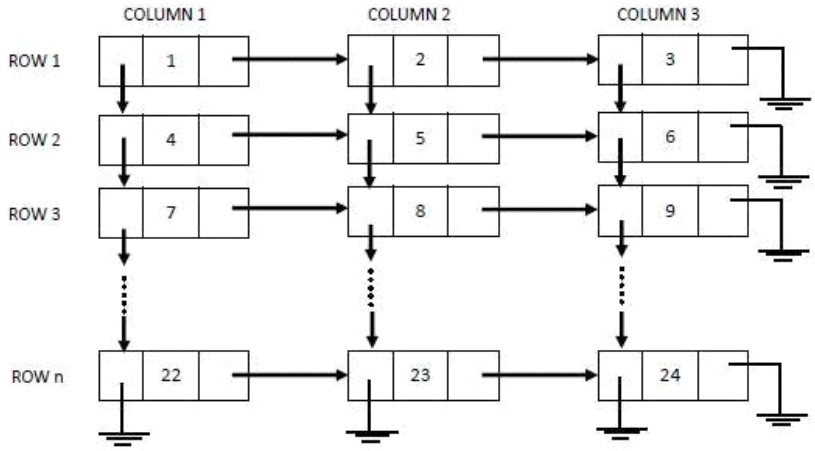
sort, *i.e.* if an object O defined as Sort <int> O, calls a member function QuickSort as O.QuickSort(A), where A is an integer array will sort the array A.

Q10. 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 ≤ 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.

Week# 6 (26th Aug to 31st Aug 2019), Week# 7 (2nd Sept to 7th Sept 2019) & Week# 8 (9th Sept to 14th Sept 2019)

Topics: Multi List/List of List/Searching/Sorting/Hashing

Q1. It is desired to store the information (1, 2, 3 ... 24) into the data structure depicted in following figure:



Create the above structure, and display its contents, row-wise and column-wise.

Q2. Given an array of *n* numbers, give an algorithm (with given constraints) for checking whether there are any duplicate elements in the array or not? Constraint 1: extra space is not a constraint, you may use as much space as you want, Constraint 2: you can use extra space of O(1).

Q3. Given an array of *n* numbers, give an algorithm (with given constraints) for finding the element which appears the maximum number of times in the array? Constraint 1: extra space is not a constraint, you may use as much space as you want, Constraint 2: you can use extra space of O(1).

Q4. Given an array of *n* numbers, give an algorithm for finding the first element in the array which is repeated. For example, in the array *A* = {3,2,1,2,2,3}, the first repeated number is 3 (not 2). That means we need to return the first element among the repeated elements.

Q5. You have been given the address of the first node of a singly linked list. Here, each node of the linked list has two fields: an integer element and a pointer of self type. It is desired to sort this linked list. Can you perform following, if yes then discuss how:

(a) We can sort the linked list using Bubble sort with constant extra space
(b) We can sort the linked list using Quick sort with constant extra space
(c) We can sort the linked list using Merge sort with constant extra space
(d) We can sort the linked list using Selection sort with constant extra space

Q6. You have been given the address of the first node of a singly linked list. Here, each node of the linked list has two fields: an integer element and a pointer of self type. A specific pattern in elements of the successive nodes of the linked list is given as follows: element in i^{th} node (where, i is 0, 2, 4, 6,

8 ...) is bigger than the elements in $(i+1)^{th}$ node, whereas it is smaller than $(i+2)^{th}$ node. Given that extra space is not a constraint (i.e. you may use as much space as you want), which one of the following sorting technique you will apply to sort the given linked list: bubble sort, merge sort, selection sort.

Q7. Hash table and hash functions in hashing are used to store the items/elements in such a way that it will be easier to find the elements later using the hash function. Hashing can be open or closed. In open hashing keys/elements are stored in linked list attached to cells/slots of the hash table whereas, all keys/elements in closed hashing are stored in the hash table itself. Collision is one of the major problems in hashing where two or more than two elements share the same hash value and hence needed to be resolved. Linear probing, quadratic probing, re-hashing, chaining, etc. are used to implement the hashing and resolve the collision. Write programs to implement hashing using (a) linear probing, (b) quadratic probing, (c) re-hashing, and (d) chaining and store following elements in each scenario: 5, 2, 15, 25, 11, 19, 65, 23, 34, 44, 64, 74, 73, 2, 15, 29, 55, 65, 26, 64, 38, 12, 16, 49, 11, 78, 65, 63, 57, 42 Further, perform following operations:

(a) Search an element 44
(b) Search an element 94
(c) Search an element 65 along with its count
(d) Remove all duplicate entries
(e) Search an element which appears maximum number of times
(f) Search two elements, X and Y whose sum is equal to user inputted number Z

Q8. Given an array of *n* elements, find two elements in the array, such that, their sum is equal to the given element *K.*

Q9. Given an array of *n* elements, find three elements in the array, such that, their sum is equal to the given element *K.*

Q10. Analyze the difference between following:

Bucket Sort and Count Sort; Bucket Sort and Quick Sort; Radix Sort and Bucket Sort

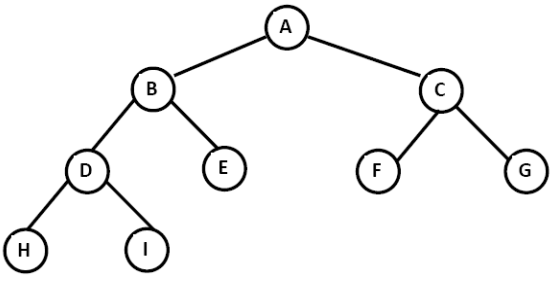
T1 is scheduled between 16th September 2019 and 21st September 2019:
Syllabus of T1: Class, Objects, Relationships, Polymorphism, Template, STL, Implementation of following: Linked List, Stack, Queue, Searching (Linear, Binary, and Median), Sorting (Bubble sort, Selection sort, Insertion sort, Quick sort, Merge sort, Bucket sort, Count sort, and Radix sort), and Hashing (Chaining and Open Addressing - Probing) with and without using STL

Week# 9 (23rd Sept to 28th Sept 2019), Week# 10 (30th Sept to 5th Oct 2019)

Topics: Binary Tree, Binary Search Tree (BST), Threaded BST (TBST)

Q1. Discuss the questions of T1 and their solutions.

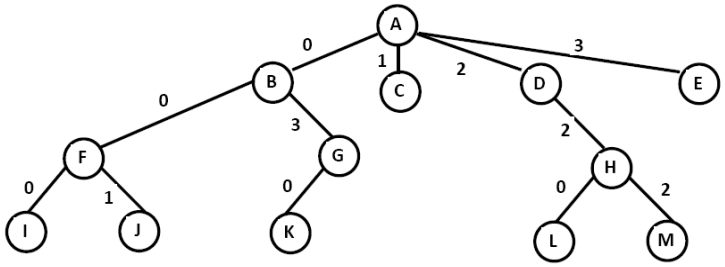
Q2. You have been given a list of characters LIST as {A, B, C, D, E, F, G, H, and I). Write a C++ program to insert these elements into a complete binary tree as shown following figure.



Also write the Recursive and Non-recursive C++ codes to traverse the binary tree as follows:

(a) In-order: traversed node sequence should be HDIBEAFCG
(b) Pre-order: traversed node sequence should be ABDHIECFG
(c) Post-order: traversed node sequence should be HIDEBFGCA
(d) Level-order: traversed node sequence should be ABCDEFGHI
(e) Spiral-order: traversed node sequence should be ABCGFEDHI

Q3. In K-ary tree, each node (except leaf nodes) can have at most K branches (***i.e.*** children). Write a C++ program to create a K-ary tree where the branches/child (at most K) present for each node is inputted by the user along with the information/data to be contained by the node. Following figure shows an example of a K-ary tree where K is 4. In this figure, the numbers associated with each edge represents the Branch No. / Child No. (Ranging between 0 and 3 because K is 4) of the parent node.



Also write the Recursive and Non-recursive C++ codes to traverse and print the node information of the K-ary tree as follows:

(a) In-order: First traverse Child [0] and Child [1] then traverse/print the root node information then traverse the Child [2] and Child [3]. For above example, printed information for this traversal is: IJFBKGCADLHME
(b) Pre-order: First print the root node information then traverse the Child [0] to Child [3]. For above example, printed information for this traversal is: ABFIJGKCDHLME
(c) Post-order: First traverse the Child [0] to Child [3] and then print the root node information. For above example, printed information for this traversal is: IJFKGBCLMHDEA

Q4. There can be maximum K branches of a node, N in a K-ary tree. Fig. 1 represents a K-ary tree (where K is 4) and Fig. 2 represents its mirror image (i.e. all the branches in original tree gets reversed connected in the mirror image). In These figures, characters at nodes represent the information stored into the node whereas, labels at edges represent the child number of a node, e.g. edge labelled as 0 between node A and node B represents the child number 0 of the node A. Use appropriate data structures and traverse the given original tree to convert it into its mirror image.

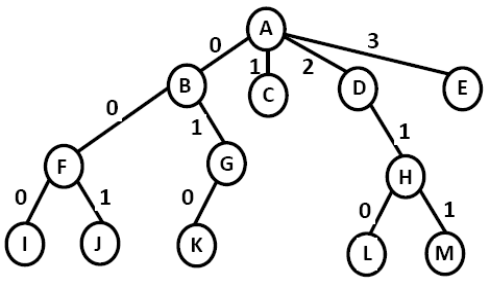


Fig. 1: Original K-ary tree

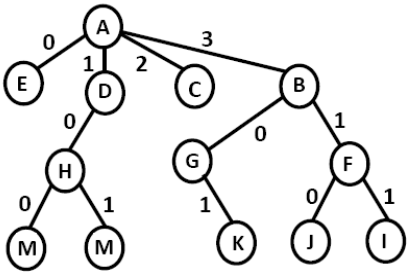


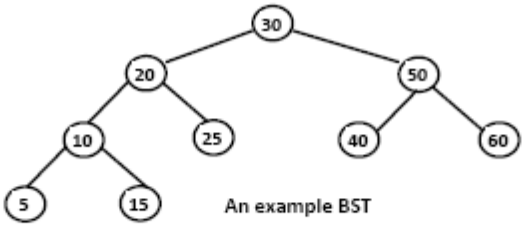
Fig. 2: Mirror image

Q5. Modify the structure of the nodes of the BST so that the BST can contain the duplicate elements and then write a program to insert following into the modified BST:

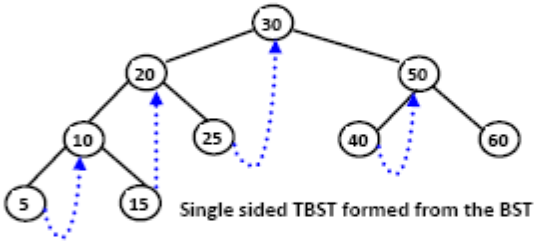
10, 20, 30, 40, 50, 40, 35, 25, 20, 15, 18, 19, 22, 27, 30, 27

Write a program to display (using In-order traversal) the elements of the modified BST

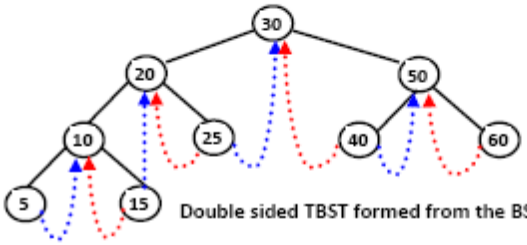
Q6. In threaded BST (TBST), the child pointers of a leaf node, N, which are NULL, are replaced (i.e. threaded) with in-order predecessor (if Left child of N is NULL) if available or in-order successor (if Right child of N is NULL) if available. Threading in TBST can be achieved in two ways, Single sided TBST, where right child pointer (which is NULL) of a leaf node, N, is replaced with the in-order successor of N; and Double Sided TBST where left child pointer (which is NULL) and right child pointer (which is NULL) of a leaf node, N, is replaced with the in-order predecessor and in-order successor of N respectively. An example of a BST, its Single sided TBST, and its Double sided TBST are shown in following figures:



An example BST



Single sided TBST formed from the BST of above example



Double sided TBST formed from the BST of above example

Write a program to create the Single sided and Double sided TBST from the given BST. After creation, write a program to delete an element/node from Single sided and Double sided TBST.

Week# 11 (7th Oct to 12th Oct 2019), Week# 12 (14th Oct to 20th Oct 2019)

Topics: AVL Tree, B Tree, B+ Tree

Q1. Construct an AVL tree by inserting following elements (one by one): 10, 20, 30, 40, 50, 45, 35, 25, 15, 5, 8, 18, 28, 38, and 48
After creation of the AVL tree, one by one delete following elements: 38, 50, and 10

Q2. You have been given an array of n elements, where elements are arranged in ascending order. Starting with first element of the array you keep on inserting the elements into an AVL tree. How many rotations will be performed while inserting all the n elements into the AVL tree.

Q3. For a given AVL tree, it is desired to (a) print all the leaf elements (i.e. elements present in leaf node where leaf nodes are those nodes which do not have any child), (b) print all the elements present in such nodes which are having only one child branch (either left child or right child), and (c) print all the elements present in such nodes which have presence of both children. Write a program to perform the mentioned tasks.

Q4. It is desired to check whether given BST is an AVL Tree or not. Write a program to perform the desired task.

Q5. It is desired to insert following elements {10, 20, 30, 40, 50, 45, 35, 25, 15, 5, 8, 18, 28, 38, and 48} into (a) B Tree of Order (double pass) 3, and (b) B Tree of Degree (single pass) 2. Construct both the B Trees by inserting the mentioned elements one by one. Further you need to delete following elements (one by one) from both B Trees: 18, 50, 25, 30, and 28

Q6. Analyze the suitability of the B+ tree over B tree.

Q7. It is desired to insert following elements {10, 20, 30, 40, 50, 45, 35, 25, 15, 5, 8, 18, 28, 38, and 48} into B+ Tree of Order 3. Construct the B+ Trees by inserting the mentioned elements one by one. Further you need to delete following elements (one by one) from the constructed B+ Trees: 18, 50, 25, 30, and 28

T2 is scheduled between 21st October 2019 and 25th October 2019 followed by Mid Semester Break scheduled between 26th October 2019 and 31st October 2019:
Syllabus of T2: Binary Tree, Binary Search Tree (BST), Threaded Tree, AVL Tree, B Tree, B+ Tree

Week# 13 (4th Nov to 9th Nov 2019)

Topics: Binary Heap

Q1. Discuss the questions of T2 and their solutions.

Q2. Construct a Max Binary Heap and Min Binary Heap using following elements: 10, 20, 30, 40, 50, 45, 35, 25, 15, 5, 8, 18, 28, 38, and 48

Q3. Analyse the performance of the Binary Heap, if it is created using (a) array and (b) binary tree (complete)

Q4. You have been given a Max Binary Heap and it is desired to convert it into a Min Binary Heap. Write a program to perform the desired task.

Q5. You have been given two Max Binary Heaps and it is desired to merge both heaps into a single heap. Write a program to perform the desired task.

Week# 14 (11th Nov to 16th Nov 2019), Week# 15 (18th Nov to 23rd Nov 2019)

Topics: Graph & Performance Evaluation

Q1. Store the Weighted Graph presented in below figure (Fig. 1) using Adjacency Matrix and Adjacency List.

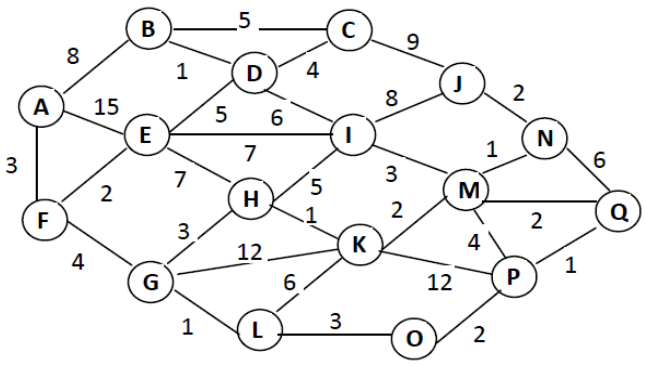


Fig. 1: A weighted graph

Q2. Using Breadth First Search, it is desired to find out the minimum number of edges required to traverse from Node A to Node P (Fig. 1 of Q1). Write a program to perform the desired task. Also discuss the list of data structures used to perform the desired task.

Q3. It is desired to find out all the paths to traverse from Node A to Node P (Fig. 1 of Q1). Write a program to perform the desired task. Also discuss the list of data structures used to perform the desired task.

Q4. It is desired to find out the minimum cost required to traverse from Node A to Node P (Fig. 1 of Q1). Write a program to perform the desired task. Also discuss the list of data structures used to perform the desired task.

Q5. Find out the Minimum Spanning Tree of the weighted graph given in Fig. 1 using Kruskal's algorithm and Prim's algorithm. Write a program to perform the desired task. Also discuss the list of data structures used to perform the desired task. Evaluate the performances of both approaches (Prim's and Kruskal's) in finding the MST.

Week# 16 (25th Nov to 30th Nov 2019)

Doubt removal week and class test.

Syllabus of End Term Examination: All topics upto T2, Heap, and Graph