Based on C++ Concepts of Class XI (C++ Revision Tour)

1. **Token** - Smallest individual unit of the program ex. Identifier, keyword.
2. **Keyword** - It is predefined reserved identifier that has special meaning e.g int,while
3. **Variable** - It is the memory location which can hold certain values. It is a sequence of alphabets & digits but the first character must be an alphabet or underscore.
4. **Data type** - It is the way to identify type of data and associated operations which can be performed on the data. Different types:- i) simple/fundamental –int, float char, double and

ii) structured/Derived – Arrays, structures, classes etc.

5. **const Keyword** – This keyword is used to declare a constant variable whose value can't be changed e.g. const int x = 5; or int const x = 5; will create an integer variable x with initial value 5 that a program cannot change.
6. **break** – the break statement is used to exit the most current loop. Syntax: break;
7. **continue:** The continue statement does the opposite of break statement; instead of exiting a loop early, continue forces the computer to perform the another iteration of the loop. Syntax: continue;
8. **typedef Keyword** – It is used to rename an existing data type.
   Syntax: typedef <existing datatype> <new name>;
9. **# define Directive** – The # define preprocessor directive creates symbolic constants.

Syntax: - #define <identifier> <replacement text>

e.g. # define PI 3.14159 will replace all occurrences of PI with value 3.14159.

2 Mark questions

Q1. **Difference between Global and local variables.**

Ans. i) Global Variable is a variable which is declared outside all functions whereas local variable is a variable which is declared within a function or within a compound statement.

ii) Global variable is accessible throughout the program whereas local variable is accessible within the function /compound statement in which it is declared.

Example: #include<iostream.h>

```
int a = 10; // Global variable(accessible to all functions)

void main()

{

        int a = 5; // local variable

        cout<<a; // display value 5 of local variable

}

void f1()

{

        cout<<a;  // display value 10 of global variable

 }
```

Q2. **Difference between OOP and Procedural Programming.**

OOP	**Procedural**
1.Emphasis on data (Object)	1 Emphasis on doing things (functions).
2. Follow bottom-up approach in program design	2. Follow top-down approach in program design
3. Data hiding feature prevents accidental change of data.	3. Presence of global variables increases chances of accidental changes in data.
4. Features like data encapsulation, polymorphism and inheritance are present.	4. Such features are not present.

Q3. **Difference between # define and const with example.**

Ans. **#define Directive** – the # define preprocessor directive creates symbolic constants.

Syntax: - #define <identifier> <replacement text>
e.g. # define PI 3.14159 will replace all occurrences of PI with value 3.14159.

const Keyword – This keyword is used to declare a constant variable whose value can't be changed e.g. const int x = 5; or int const x = 5; will create an integer variable x with initial value 5 that a program cannot change.

Q 4. **Purpose of using typedef command in C++. Give e.g.**

Ans. **typedef Keyword** – It is used to rename an existing data type.

Syntax: typedef <existing datatype> <new name>;

e.g typedef int age; age boy girl; //boy and girl variable will be of integer type

Q5. **Difference between Actual Parameter and Formal Parameter**.

Ans. **Actual Parameters**- These are the arguments that appear in function call statement.

Formal Parameters - These are the arguments that appear in function definition.

e.g. # include <iostream.h>

float sum (int x, float N); // func. Prototype declaration of function called sum

```
void main()
{
        float p,q;
        int r;
        cin>>r>>p;
        q = sum(r, p); // Actual parameters
}
float sum (int x, float N)                    // formal parameters
{……}
```

Q6. **Differentiate between call by value and call by reference with e.g.**

Ans.

CALL BY VALUE	**CALL BY REFERNCE**
This method copies the value of actual parameters into formal parameters. The function creates its own copy of argument values & then uses them; so any changes made to formal parameters are not reflected back to calling function.	This method passes a reference /alias to the original variable / actual parameters. The same variable value can be used by any two names: the original (actual variable) and reference (formal parameter). Thus all changes made to formal parameters are reflected back to the calling function.

1(b) **Header file (1 mark)** Can come in 2 forms.

One form can be predefined functions would be given and name of header file containing that function is to be given:

(I) strlen () , strcat() - string.h

Note: - All functions starting with str are declared in header file **string.h**

(II) isupper(), islower (), toupper(), tolower(), isalpha(), isdigit (), isalnum () - ctype.h

Note: - All functions in which type of character is being checked are in **ctype.h**

(III) sqrt(), abs(), frexp(), ceil(), fabs(), - math.h

Note: - All mathematical functions are declared in math.h

(IV) setw() - iomanip.h
(V) gets(), puts() - stdio.h
(VI) clrscr(), getch() - conio.h
(VII) random(), randomize(),atoi( ),itoa( ) - stdlib.h
(VIII) open(), close() - fstream.h

(IX) exit() - process.h, stdlib.h

II form of question can be – Code snippet [part of program] would be given

e.g. void main()

{ char string[ ] = “Peace”;

cout<<setw(20)<<string;

getch(); }

Ans. iostream.h, iomanip.h , conio.h

1(c) **Error finding (2 marks)**

Write the correct version of the program with each correction underlined and then explaining the correct statement corresponding to the correction made.

1(d) **Output Question of 2 marks**

1(e) **Output Question of 3 marks**

1 (f) **Question based on random function (2 marks)**

Can come in following forms:-

(I) In the given program if value of N given by user is 20, what maximum & minimum values the program could possibly display?

```
#include<iostream.h>     #include<stdlib.h>
void main( )
{
    int N,guess;
    randomize();
   cin>>N;
   guess = random(N-10)+10;
   cout<<guess<<endl;
}
```

Ans. random(N) always generates values from 0 to N-1 so guess can take value. Random (N-10)+10 => random(20-10)+10 => Random(10) +10

Now random (10) can generates from 0 to 9 so first part of equation i.e. random (10)can have min value 0 and 10 is added to it. Guess can have min value = 0+10 = 10. Guess can have max value = 9+10 = 19

(II) in the following program find correct possible output(s) from the given options:-

```
#include<stdlib.h>
#include<iostream.h>
void main()
{
        randomize();
        char person[ ] [10] = {" nancy", sweety","esha", "usha"};
        int togo;
        for(int i = 0; i <3;i++)
        {
        togo = random(2)+1;
        cout<<person[togo]<<":";
        }
}
```

OUTPUTS (i) sweety:esha:sweety iii) sweety:esha:usha

ii) nancy:sweety:esha iv) sweety:esha:esha

Ans. Starting with for loop when i = 0 then random (n) generates values from 0 to n-1.

Assuming random (2) generates value 0 then togo can have value 1 so cout<<person [1] will display string sweety so option (ii) ruled out. Coming to for loop again value of I = 1 then again togo can take value 2 or 1 so string displayed can be Esha or sweety. Now I = 2 similarly same string can be displayed so (iii) option is also ruled out because usha is at index 3 and value of togo would never be 3.

Final options are (i) and (iv)

2(a) **Can come from any of the following concepts [2 marks]**

OBJECT ORIENTED PROGRAMMING (OOP) CONCEPTS

PARADIGM – It means organizing principle of a program. It is an approach to programming.

PROCEDURAL PARADIGM- In this, the emphasis is on doing things i.e. the procedure or algorithm. The main focus is on functions rather than data. This paradigm does not model real world well.

OBJECT ORIENTED PARADIGM –It models the real world well and overcomes the short comings of procedural paradigm. It views a problem in terms of objects i.e. main focus of this approach is given to objects and thus emphasis is on both procedures as well as data.

OBJECT- It is an identifiable entity with some characteristics and behavior. It is an instance of a class.

CLASS- A class represents a group of objects that share common properties, behavior and relationships.

DATA ABSTRACTION-It refers to act of representing essential features w/o including background details or explanations.

ENCAPSULATION- The wrapping up of data and its associated functions into a single unit called class is known as encapsulation. It is the way to implement abstraction.

INHERITANCE- It is the capability of one class of things to inherit the capabilities or properties from another class. The class whose properties are inherited is called is known as base class or super class and the class that inherits the properties of another class is called the derived class or sub class.

POLYMORPHISM- It is the ability for a message or data to be processed in more than one form. It is the property by which same message can be sent to objects of several different classes.

MODULARITY – It is the property of a s/w system or program that has been decomposed into different modules or functions.

2(b) Question based on Constructor & Destructor

CONSTRUCTOR – Constructor is the member function of the class having the same name as that of the class name. It is automatically called by the compiler whenever an object of the class is created.

DESTRUCTOR- It is the reverse of constructor. It is called when the scope of the object is over.

DEFAULT CONSTRUCTOR – A Constructor that accepts no parameters.

COPY CONSTRUCTOR- A copy constructor is a constructor that is invoked when an object is defined and initialized with another object. A copy constructor takes the following form classname (classname &)

e.g. Given following C++ code, answer the questions that follows:-

```
class readbook
{ public:
readbook()                          // function 1
{   cout <<"open the book"<<endl; }
readbook()                          // function 2
{   cout <<"reading chapter one"<<endl; }
~readbook()                         // function3
{   cout <<" close the book"<<endl; }
```

In OOP what is function 1 referred as & when does it get called?

Ans. Constructor and is called when an object of the class is created.
Function 3 is referred to as destructor and is called when object goes out of scope.

DATA STRUCTURE

DATA STRUCTURE A mathematical and logical model of data is known as data structure.

LINEAR DATA STRUCTURE- The data structure in which each element has access to maximum of one predecessor element and maximum of one successor element e.g. stack, queue.

NON- LINEAR DATA STRUCTURE- The data structure in which each element has access to any number of predecessor element and any number of successor element e.g. stack, queue.

STATIC DATA STRUCTURE - The data structure in which number of elements is fixed. E.g. arrays [Memory is allocated at compile time]

DYNAMIC DATA STRUCTURE- The data structure in which number of elements is not fixed. E.g. Linked List [Memory is allocated at execution time]

C++ Implementation (1-D Arrays)

Function to transverse the array arr

```
void Transverse (int arr [ ], int l)
{                                       // l-> size of array
for (int c=0;c<l;c++)           // arr -> integer array
cout <<arr[c]<<endl;
}
```

Function to read elements of array arr

```
void  input(int arr [ ], int l)
{                                       // l-> size of array
for (int c=0;c<l;c++)           // arr -> integer array
cin>>arr[c];
}
```

Function to search for an element from an array 'arr' by Linear search

```
void  Lsearch(int arr [ ], int l)
{
        int data, found =0; c=0;                    // l-> size of array
        cout<<"Enter the data to be searched";                  // arr -> integer array
        cin>>data;
        while(c<l && found!=1)
        {
          if(arr[c] = = data)
          found++;
          else c++:
        }
        if(found)
        cout<<"Data found at :"<<c<<endl;
        else
        cout<<"Data not found";
}
```

Function to sort the array 'arr' by Bubble Sort

```
void  BubbleSort(int arr [ ], int l)
{
        for (int i = 0; i<l-1; i++)
        for (int j= 0;j<l-i-1;j++)
        {
                if(arr[j] > arr[j+1])
                {
                int temp = arr[j];
                arr[j] =arr[j+1];
                arr[j+1] = temp;
                }
        }
}
```

Function to sort the array 'arr' by Insertion Sort

```
void Insertionsort(int arr [ ], int l)
{
        for (int i = 0; i<l; i++)
        {
                int temp = arr[i];
                j = i-1;
                while( temp< arr[j] && j > =0)
                {
                        arr[j+1] = arr[j];
                        j--;
                }
                arr[j+1] = temp;
        }
}
```

Function to sort the array arr by Selection Sort

```
void Selectionsort(int arr [ ], int l)
{
        for (int i = 0; i<l-1; i++)
        {
                int small = arr[i];
                int pos = i;
                for (int j = i+1; j<l; j++)
                {
                        if ( small > arr[j])
                        {
                                small = arr [j];
                                pos = j;
                        }
                        int temp = arr[i];
arr[i] = arr[pos];
arr[pos] = temp;
}
```

to search for an element by Binary search

```
// assuming that array is presorted in increasing order
void  Binarysearch( int arr [ ], int l, int val)
{
        int lb=0, ub =l-1, mid,found=0;
        while( lb = <= ub && found == 0)
        {
                mid = (lb+ub)/2;
                if (arr[mid] == val)
                        found ++;
                else if ( arr[mid]<val)
                        lb = mid +1;
                else
                        ub = mid -1;
        }
        if (found = =1)
                cout <<" found at"<<mid<<endl;
        else
                cout<<"not found"<<endl;
}
```

Function to merge X and Y arrays of length M & N

```
// assuming that array is presorted in increasing order
void  merge ( int X[ ], int Y[ ], int arr[ ], int m,int n, int &l)
{
    int i =0, j=0, k=0;
    L= m+n;
    while(i<m && j<n)
    {
       if ( X[i]<Y[j])
       arr [k++]= X[i++];
       else
       arr[k++]= Y [j++];
```

```
    }
    while (i <m)
    {
        arr[k++] = X[i ++];
    }
    while (j < n)
    {
        arr [ k++] = Y [ j ++];
    }
}
```

3 (c) DYANAMIC ALLOCATION [4 MARKS] (BASED ON LINKED STACK & LINKED QUEUE)

LINKED QUEUE (Insert Operation)

```
struct NODE
{
   char name [30] ;
   NODE * link;
};
NODE * rear,* front;
```

```
void insert ( )
{
    NODE *temp;
   temp = new NODE;
   cout<<" enter data of new node";
   gets(temp -> name);
   temp-> link = NULL;
   if (rear == NULL)
   {
      rear = front = temp;
   }
   else
   {
      rear -> link= temp;
      rear = temp;
   }
}
```

LINKED QUEUE (Delete Operation)

```
struct NODE
{
   char name [30] ;
   NODE * link;
};
NODE * rear,* front;
```

```
void delete( )
{
  NODE *ptr = front;
   if (ptr = = NULL)
   {
     cout <<"underflow";
     exit(1);
   }
   else if (front = = rear)
   {
      delete ptr;
      f ront = NULL;
      rear = NULL;
   }
   else
   {
      front = front-> link;
     delete ptr;
   }
}
```

LINKED STACK (PUSH Operation)

```
struct NODE
{
   float data;
   NODE * link;
};
```

```
void PUSH(NODE * top, float num )
{
    NODE  *nptr = new NODE;
    nptr -> data = num;
    nptr -> link = NULL;
    if (top == NULL)
    top = nptr;
    else
    {
       nptr -> link = top;
       top = nptr;
    }}
```

LINKED STACK (POP Operation)

```
struct book
{
   int Bno;
   char bname[20];
   book * next;
};

void POP(book * top)
{
   book  *nptr = top;
   if (ptr == NULL)
   cout<<"underflow";
   else
    {
      cout<<"element being deleted is\n";
      cout<<"bno"<<top->bno;
      cout<<"bname"<<top->bname;
    }
   top = top-> next;
   delete ptr;
 }
```

3 (b) ADDRESS CALCULATION

(I) (ROW WISE) B + w [C (I – Lr) +(j- Lc)]

Where B = Base Address, w = word size , C = number of columns[main array]
Lr = lower bound for row
Lc = lower bound for column, i,j = subscript whose position or address is to be calculated

(II) (COLUMN WISE) B + w [(I – Lr) + R (j- Lc)]

where R =number of rows[main array]
i,j = subscript whose position or address is to be calculated

if array is given in form
A[15……20,13…..25]
Then lr = 15
Lc – 13
No. of rows(R) = 20-15+1 = 5+1 = 6
Ur-lr +1
No. of columns(C) = 25-13+1 = 12+1 =13
Uc= 25
Lc = 13
If an array is given in form

A[13][20]
Then lr = 12
Ur = 12
Lc =0
Uc= 19
R= No. of rows = 13
C= No. of columns = 20
Use given values & substitute in the formula in row major or column major acc. to the given question.

TWO DIMENSIONAL ARRAY(C++ IMPLEMENTATION)

FUNCTION TO READ THE ARRAY A

```
void read( int A[3][20],int M, int N)
{
    for (int R = 0; R<M; R++)
     for(int C =0;C<N;C++)
    {
       cin>>A[R][C];
    }
}
```

FUNCTION TO FIND SUM OF 2-D ARRAYS A & B

```
void ADD( int A[ ][20], int B[ ][20], int C[ ][20],int N, int M)
{
  for (int R = 0; R<N; R++)
  for(int c=0;c<M;c++)
   {
     C[R][C] = A[R][c] + B[R][c];
   }
}
```

FUNCTION TO MULTIPLY OF 2-D ARRAYS A & B OF ORDER N X L & L X M

```
void MULTIPLY( int A[ ][20], int B[ ][20], int C[ ][20],int N,int L, int M)
{
  for (int R = 0; R<N; R++)
  for(int c=0;c<M;c++)
  {
    C[R][c] = 0;
    for (int T = 0; T<L; T++)
    C[R][C]+ = A[R][T] * B[T][c];
  }
}
```

FUNCTION TO FIND AND DISPLAY SUM OF ROWS OF 2-D ARRAY A

```
 void SUMROWCOL( int A[ ][20],int M,int N)
```

```
  {
      for (int R = 0; R<M; R++)
      {
          int sumR = 0;
          for(int C =0;C<N;C++)
          {
            sumR + = A[R][C];
          }
          cout <<"Row["<<R<<"]"<<sumR<<endl;
      }
  }
```

FUNCTION TO FIND AND DISPLAY SUM OF COLS OF 2-D ARRAY A

```
void SUMROWCOL( int A[ ][20],int M,int N)
{
      for(int C =0;C<N;C++)
      {
            int sumC = 0;
             for(int R =0;R<M;R++)
            {
                sumC + = A[R][C];
            }
            cout <<"column["<<C<<"]"<<sumC<<endl;
      }
}
```

FUNCTION TO FIND THE SUM OF DIAGONAL ELEMNTS OF SQUARE MATRIX A

```
void DIAGONAL( int A[20][20],int N,int Rdiag,intLdiag)
{
   Rdiag =0 , Ldiag =0;
   for (int I = 0; I<N; I++)
   {
       for (int J = 0; J<N; J++)
       {
          if(I = =J)
          Rdiag + = A[I][J];
          if (I+J = =N-1)
          Ldiag + = A[N-I-1][I];
       }
   }
}
```

FUNCTION TO CREATE TRANSPOSE OF A 2-D ARRAY A

```
void transpose( int A[20][20],  int B[20][20],int N,int M)
{
     for (int R = 0; R<M; R++)
    {
        for (int C = 0; C<N; C++)
        B[R][C] = A[C][R];
    }
}
```

FUNCTION TO DISPLAY CONTENT OF A 2-D ARRAY A

```
void display( int A[20][20], int N,int M)
{     for (int R = 0; R<M; R++)
    {
         for (int C = 0; C<N; C++)
         {
              cout<<A[R][C]<<'\t';
         }
          cout<<endl;
     }
}
```

Ques: If INPUT ARRAY is 1,2,3,4,5,6

```
void FUNC( int arr[ ], int size)
{       int  a [20][20] , i  , j;
        for (int i = 0; i<size; i++)
           for (int j = 0; j<size; j++)
                   {
                      if(i+j) >= size)
                      A[i][j] = 0;
                      else a[i][j] = arr[j];
                   }
        for (int i = 0; i<size; i++)
          {
             for (int j = 0; j<size; j++)
                      cout<<a[i][j]<<"";
             cout<<endl;
          }
 }
```

OUTPUT ARRAY should be

1	2	3	4	5	6
1	2	3	4	5	0
1	2	3	4	0	0
1	2	3	0	0	0
1	2	0	0	0	0
1	0	0	0	0	0

FUNCTION TO INSERT DATA AT POS IN THE ARRAY ARR

```
void INSERT( int arr[ ], int &L,int pos,int data)
 {
    if(l< max)
    {
          for (int C = L; C<POS; C--)
         {
              arr[C] = arr[C-1];
         }
          arr[pos] = data;
```

```
            l++;
    }
    else
       cout<<"array is full"<<endl;
}
```

FUNCTION TO DELETE DATA AT POS IN THE ARRAY ARR

```
void DELETE( int arr[ ], int &L,int pos)
{
   for (int C =pos; C<L-1; C++)
   {
      arr[C] = arr[C+1];
    }
    L--;                                    // after deletion of an element decrease L
}
```

DATA FILE HANDLING

FILE:- Bunch of bytes stored on some storage media. **STREAM:-** Flow of data.

4(a) Fill up with seekg( ),seekp( ) , tellp ( ), tellg( ) ,write( ) and read( ) functions

These function are member functions of different (header files or class)

<ofstream.h>	**<ifstream.h>**
tellp() it returns position of put pointer in terms of no. of bytes	**tellg() it returns positions of get pointer in terms of no. of bytes**
seekp() it moves put pointer to required position.	**seekg() it is used to move get pointer to required position.**
write() used to write a block of data on a file.	**read() used to read a block of data from a file.**

And the member function can be called through object only

seekg() and seekp() help in attaining random access in file.

Syntax of read() :- file object.read((char*) &object , size of (object);

Syntax of write():- file object.write ((char*) &object, size of(object):

ofstream fout("abc.dat",ios:: binary);

ifstream.h fin ("abc.dat",ios:: binary);

Object are fout(writing to file) and fin (reading from file)

fout.seekp(……)	fin.seekg(……)
fout.tellp(……)	fin.tellg(……)
fout.write(……)	fin.read(……)

Statement to replace the file pointer at the starting of record.

File.seekg (- 1) * size of (ob), ios : : cur);

Statement to write the record

File.write ((char*) &ob, size of (ob));

Statement to place the write pointer

File.seekp(recordno*sizeof(p), ios::beg);

Statement to place the read pointer

File.seekp(o,ios::end);

Tellg function returns the location of file pointer in bytes.

K= file.tellg ( );

b) Write the function to finds the occurrence of any word in a file "txt"

```
void word count ( )
{
        ifstream fin (" ------ txt");
        char word [30];
        int count = 0;
        Write (! fin. eof () )
```

```
{
cin >>word;
if (strcmp("the",word) = = 0)
count + +;
}
fin.close ( );
cout<<" The no of occurrence "<<count;
}
```

TEXT FILE	BINARY FILE
Information stored in ASCII characters	Information stored in same format as it is stored in binary.
Some internal translations take place.	No internal translations take place. Faster to read & write than text files.

Files can be opened by:-

- Using constructor function of stream class.
- Using the function open()

ios::in	Opens file for reading i.e. in input mode.
ios::out	Opens the file for writing i.e. in output mode
ios::app	Opens the file for writing without destroying previous data
ios::trunc	This causes the contents of pre existing file by the same name to be destroyed and truncates file to zero length.
ios::nocreate	Causes open( ) function to fail if file does not already exist. It will not create a new file with that name. If file exists it gets opened.
ios::noreplace	If the file does not exist, a new file gets created but if the file already exists, the open () fails
ios::binary	Opens file in binary mode

Note:- By default files are opened in text mode. When a file is opened in text mode, various translations may take place such as the conversion of carriage return into new line. No translations occur in files opened in binary mode.

DETECTING EOF(end of file)

Using eof( ) function:-

Proto type is int eof( );

It returns non zero when the end of file hasd been reached, otherwise it returns zero.

Q5. (a) DATABASE CONCEPTS(2 marks)

DATABASE:- It refers to collection of logically related data.

DATA REDUNDANCY:- Duplication of data.

DATA INCONSISTENCY:- Multiple mismatching copies of same data represent data consistency

DATA SECURITY:- Protection of data against accidental or intentional disclosure to unauthorized persons or unauthorized modification or destruction.

DATA PRIVACY:- Rights of individuals and organizations to determine for themselves when, how and to what extent information about them is to be transmitted to others.

DATA INDEPENDENCE:-Ability to modify a scheme definition in one level without affecting a scheme definition in the next higher level.

PHYSICAL DATA INDEPENDENCE:- Ability to modify a scheme followed at the physical level without affecting the scheme followed at conceptual level.

LOGICAL DATA INDEPENDENCE:-Ability to modify a conceptual scheme without causing any changes in the scheme followed at the view level or external level.

RELATIONAL DATA MODEL – The data model wherein data is organized into tables called relations. Relationship among multiple tables is established on the basis of common column.

HIERARCHIAL DATA MODEL – The data model wherein data is represented in the form of parent- child records. The dependent data is known as child record.

NETWORK DATA MODEL – Relationship among data records are represented by links or pointers.

RELATION – A table having non-empty atomic values with unordered rows & columns.

DOMAIN – A pool of values where from a field can draw values is called domain.

TUPLE- A row in a relation is called tuple.

ATTRIBUTE – A column in a relation is called attribute.

DEGREE - No. of attributes (columns) in a relation.

CARDINALITY - No. of tuples (rows) in a relation.

VIEW – A virtual table that does not really exist in its own right but is instead derived from one or more underlying base tables.

PRIMARY KEY – A set of one or more attributes that can uniquely identify tuples within the relation.

CANDIDATE KEY – All attribute combinations inside a relation that can serve as primary key.

FOREIGN KEY – A non- key attribute whose values are derived from primary key of some other table.

STRUCTURED QUERY LANGUAGE (SQL)

SQL- It is a language that enables us to create and operate on relational databases which are sets of related information stored in tables. It is a standard language for communicating with RDBMS(Relational Database Management System) from any tool. SQL allows us to communicate with the database and has following advantages:-

1. Efficient
2. Easy to learn & use
3. Functionally complete

INTEGRITY CONSTRAINTS – SQL provides many constraints which are used to enforce rules at table creation level whenever row is inserted, updated or deleted from the table.

A Constraint is a condition or check applicable on a field or set of fields.

1. **COLUMN LEVEL OR COLUMN CONSTRAINT** – It references to a single column only. Any type of integrity constraint can be defined at column level.

2. **ROW LEVEL OR TABLE CONSTRAINT** – It references to one or more columns and is defined separately from the definitions of the columns in the table. When a constraint is applied on a group of columns of the table, it is called table constraint. Any type of integrity constraint can be defined at table level except NOT NULL constraint.

DIFFERENT CONSTRAINTS

1. **NOT NULL Constraint** – This Constraint ensures that the null values (empty values) are not permitted for a specific column. This Constraint can be specified at column level not at the table level. e.g. the following example defines field TNO to be NOT NULL , so field or column TNO does not accept empty values.

   create table teacher (TNO Number(5) NOT NULL, TName varchar(20), Taddress varchar(25),

   DeptNo number(5), DOJ date);

2. **UNIQUE CONSTRAINT**- This constraint ensures that no two rows have the same value in the specified column(s). This constraint can be applied only to columns that have also been declared NOT NULL. For e.g. UNIQUE constraint applied on TNO of teacher table ensures that no two rows have the same TNO.value

   create table teacher (TNO Number(5) NOT NULL UNIQUE , TName varchar(20), Taddress varchar(25),

   DeptNo number(5), DOJ date);

3. **PRIMARY KEY CONSTRAINT**- This constraint declares a column as the primary key of the table. This constraint is similar to unique constraint except that only one column (or one group of columns) can be applied in this constraint. The primary keys cannot allow NULL values, thus this constraint must be applied to columns declared as NOT NULL. The PRIMARY KEY constraint is a column or set of columns that uniquely identifies each row in a table.

   create table teacher (TNO Number(5) NOT NULL PRIMARY KEY, TName varchar(20),

   Taddress varchar(25),DeptNo number(5), DOJ date);

4. **DEFAULT CONSTRAINT-** A default value can be specified for a column using the DEFAULT CLAUSE. When a user does not enter a value for the column (having default value). Automatically the defined default value is inserted in the field for e.g. **create table teacher (TNO Number(5) NOT NULL PRIMARY KEY, TName varchar(20), Taddress varchar(25),DeptNo number(5) Default = 10);**

   In the above e.g. if no value is provided for DeptNo then the default value 10 will be entered in filed **DeptNo**.

5. **CHECK CONSTRAINT**- This constraint limits values that can be inserted into a column of the table. This constraint explicitly defines a condition that each row must satisfy. A single column can have multiple CHECK constraints that reference the column in its definition .There is no limit to the no of CHECK constraints that we can define on a COLUMN.
   e.g. Create table teacher (TNO Number (5) NOT NULL PRIMARY KEY, TName varchar (20), Taddress varchar (25), DeptNo number (5), Salary Number (10) CHECK (Salary > 10000));

   The above statement ensures that the value inserted for salary must be greater than 10000.

   Applying Table Constraint- When a constraint is applied on a group of columns of the table it is called table constraint.

   Create table teacher (TNO Number (5) NOT NULL PRIMARY KEY, TName varchar (20), Taddress varchar (25),DeptNo number (5) default=10,Salary Number (10) CHECK (Salary > 10000), UNIQUE (Tno,Tname));

   The above example defines a table constraint unique that accepts values when both TNO and Tname are unique.

SQL COMMANDS (CONTD.)

1. CREATE TABLE

2. DESCRIBE : To describe the structure of the table. e.g describe teacher;

3. **ALTER TABLE**-This command is used to change the table. It is used to change the definitions of existing tables. We can add new column, change the datatype of column or drop any constraint using alter command.

i) **ADD Clause**- Using ADD Clause we can add new column into the table e.g. to add a new column called PHONE into table Teacher we have to write:

Alter Table Teacher ADD (PHONE Number(10));

ii) **MODIFY CLAUSE** – The modify keyword is used to modify the definition of an existing column.

ALTER TABLE TABLENAME MODIFY (column datatype);

e.g. **ALTER TABLE teacher MODIFY (Phone Number (12);**
The size of the field Phone has been changed from 10 to 12.

iii) **DROP CLAUSE**-It is used to remove a column.
eg.- ALTER TABLE teacher DROP COLUMN PHONE;

4. DROP TABLE COMMAND -This is used to remove the table physically from the database. But before dropping a table it is necessary that all its rows must be deleted. Rows of a table can be deleted using delete command.

DROP TABLE TABLENAME;

5. CREATE TABLE COMMAND- A view is a virtual table with no data but can be operated like any other table. It is like a window through which we can view the data of another table which is called base table. Create view command is used to create a view.

e.g. CREATE VIEW TAXPAYEE AS SELECT * FROM EMPLOYEE WHERE GROSS>80000;

The view called **TAXPAYEE** will be having details of those employees who have gross more than 80000. Now this view can be used just like any other table.

6) DROP VIEW COMMAND- To delete a view from the database the drop view command is used.

For e.g. DROP VIEW **TAXPAYEE**;

It will remove view called **TAXPAYEE** from the database. When a view is dropped, it does not cause any change in its base table.

All above 6 commands come under DDL, They are create table, create view, alter table, drop table, drop view , describe.

Next comes DML commands- It includes **i) Insert into ii) Select iii) Delete iv) Update**

1. **INSERT INTO** –The rows or tuples are added to relations or tables using INSERT command. It is used to add a new record to the end of an existing database. The new record includes data specified with the INSERT INTO command.

E.g. INSERT INTO teacher values (10001,'abc',90001,50000);

All character strings must be enclosed in single or double quotation marks.

Syntax is :- INSERT INTO tablename values(val1, val2,……valn);

2. UPDATE Command – It is used to change the values of the table . It is used to update rows and we can also use where clause with update command

3. **DELETE** –This command is used to delete rows of a table. This command deletes the entire row not a specific column.

   Syntax:-DELETE from table name where <condition>;

   The where clause is optional. We can use where clause to delete rows which satisfy condition given in where clause. To delete all rows write DELETE FROM TABLE NAME;

4. **SELECT COMMAND**- This command is used to retrieve information from one or more databases. The select command of SQL lets us to make queries on the database. A query is a command that is given to produce certain specified information from the database table(s). A select command can be used in various ways and combinations. It can be used to retrieve subset of rows or columns from one or more tables.

   Simplest form of select is **select <column name > from tablename ;**

   eg. To display information of column TNO of table teacher the command would be like

   Select TNO from teacher;

To display all rows of teacher table write:: **Select * from Teacher;**

The above command would display all the rows with all column names specified in table called teacher.

The select statement consists mainly of 3 clauses

i) **select clause** – It lists the column to be displayed

ii) **From clause** – It specifies table name from which data is to be displayed.

iii) **Where clause** - It specifies rows which are to be selected for output. This clause allows specifying condition that can be either **true** or **false** for any row of the table.

e .g. Select TName, salary from teacher where salary =90000;

It will show TName and salary rows whose salary is equal to 90000.

We can use **Relational as well as Boolean operators** in where clause.

Relational operators :: = (equal to), > (Greater than), < (Less than) , >= (Greater than equal to) ,

<=(less than equal to) , < >(Not equal to)

Boolean operators :: AND ,OR, NOT. Boolean operators are used to combine one or more conditions.**Eg . Select TNo, TName from Teacher where Dept_No. = 90001 AND Salary > 80000;**

It will display all rows where value of dept_no is 90001 and salary > 80000 with two columns TNo &TName.

Using Special Operators in Conditions

i) **IS NULL Clause** -> When we have to search the column whose value is NULL (it indicates missing values) in the table then we use NULL clause.

Syntax:- Select columnname from tablename where columnname IS NULL;

e.g. Select * From Teacher Where Salary IS NULL;

It will display all rows where value of salary is missing or is null with all column values.

ii) **DISTINCT Clause**-> It is useful to remove the duplicate rows in the output. The result obtained in the query using DISTINCT CLAUSE does not contain duplicate rows. **Syntax:- Select Distinct Columnname from tablename;**

iii) **ALL Clause**-> The ALL clause is same as that when we don't specify DISTINCT. The ALL clause retains the duplicate rows. The form is as: **Select all columnname from tablename;**

iv) **BETWEEN Clause**-> When we want to specify the range of the column then we have to use the BETWEEN CLAUSE. The BETWEEN operator is inclusive i.e. values present on the boundary are also including in the result.

Syntax:- SELECT colname1, colname2 FROM tablename WHERE salary BETWEEN 20000 AND 35000;

V) **IN Clause** – The IN clause is used to specify condition based on a list. The IN operator selects values that matches any value present in the list of values. The form is as:

Select colname1, colname2 from tablename where colname IN (Value1, Value 2, -----Value);

e. g. Select TNO, T Name from Teacher where TNO IN (10001, 10002, 10003);

It will display all rows whose TNO is 10001 or 10002 or 10003.

(vi) **Like Clause** - SQL provides a string matching pattern. Like can be applied for string matching pattern. Like can be applied to character fields. The Like searches a character field to see if part of it matches a string.

SQL provides 2 wild card characters.

(1) % (percent) - for matching any substring % stands for multiple characters.
(2) _ (Underscore) - for matching fixed no of character._ stands for single character.

The like clause is used to select rows which are matching wild cards pattern. Patterns are case sensitive.
E g. Select TNo, TName

From Teacher

Where TName like R%;

The above statement will display TNo and TName of all teachers whose name starts with alphabet R.

Select TNo, TName

From Teacher

Where TName like '_ a %';

The above statement will display TNo and TName of all teachers whose name has 'a' as second character.

(vii) **NOT IN Clause**– Opposite of IN clause

e.g. Select TNO, TName

from teacher

where TNO NOT IN (T01, T03);

It will display TNO and TName of all teachers except whose TNo is T01 or T03.

4. **UPDATE Command** – It is used to change the values of the table . It is used to update rows and we can also use where clause with update command.

Syntax: Update <tablename>

set column = Value1

where column = value ;

E g. Update teacher

set salary = 20000

where TNo= 10001;

5. **GROUP BY** CLAUSE:- Groups the rows of the table by columns that have same values.

Syntax:- SELECT <COL1>, <COL2>

FROM <tablename>

GROUP BY <COL 3>;

HAVING CLAUSE :- It is used with group by clause.

Syntax:- SELECT <COL1>, <COL2>

FROM <tablename>

GROUP BY <COL 3>

HAVING <CONDITION>;

ORDER BY :- It is used to sort a query in a specific order.

Syntax:- SELECT <COL1>, <COL2>

FROM <tablename>

ORDER BY <column name>;

GROUP FUNCTIONS

1. avg(column name) Average function => It calculates average value of a column.

2 Max( )

3. min( )

4. sum( )

5.count( ) to count the non-null values in a column

6. count(*) to count the total number of rows in atable.

BOOLEAN ALGEBRA

6(a) By means of truth table, demonstrate the validity of the following postulates/laws of Boolean algebra (any one)

a. **Associative Law** $X + (Y + Z) = (X + Y) + Z$

$X. (Y. Z) = (X. Y). Z$

b. **Distributive Law** $X(Y + Z) = XY + XZ$

$X + Y.Z = (X + Y). (X+Z)$

c. **Commutative Law** $X + Y = Y + X$

$X.Y = Y.X$

d. **Absorption Law** $X + XY = X$

$X(X + Y) = X$

e. **Idempotence Law** $X + X = X$

$X.X = X$

f. **Involution Law** $(X')' = X$

g. **De Morgan's Theorem** $(X + Y)' = X'.Y'$

$(X.Y)' = X' + Y'$

Proof of De Morgan's Theorem using Truth Table

X	Y	X+Y	(X+Y)'	X'	Y'	X'.Y'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Since the columns of (X+Y)' and X'.Y' are identical,]

Hence first law of De Morgan's Theorem is proved. Also second law is dual of first law so it is also proved.

i.e.$(X.Y)' = X' + Y'$ Hence Verified

Prove Algebraically

(i) $XY + YZ+YZ' = Y$

➔ $XY + Y (Z+Z')$ (BECAUSE $Y(Z+Z') = YZ+YZ'$ acc to Distributive Law)

➔ $XY + Y (1)$ (BECAUSE $(Z+Z') = 1$) acc to Complementary Law)

➔ $XY + Y$ (BECAUSE $(Y.1) = Y$) acc to Properties of 0 & 1)

➔ $Y(X + 1)$ (BECAUSE $X +1 = XY+Y$ acc to Distributive Law)

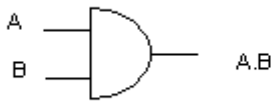
➔ $Y .1$ (BECAUSE $(X+ 1) = 1$ acc to Properties of 0 & 1)

➔ Y (BECAUSE $(Y. 1) = Y$ acc to Properties of 0 & 1)

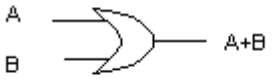
➔ R.H.S. Hence Proved

6 (b) **Interpret the following logic circuit as Boolean Expression**

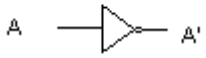
i) AND



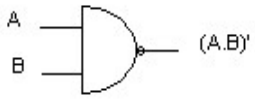
ii) OR



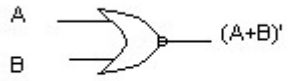
iii) NOT

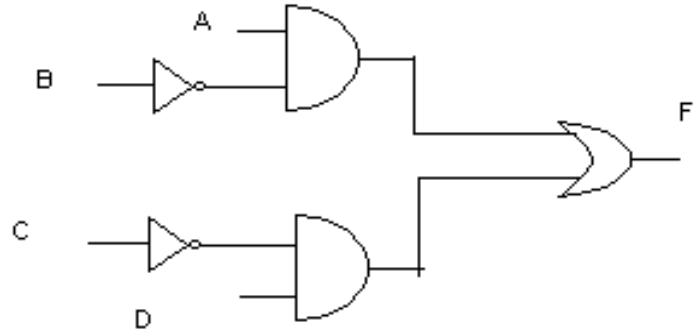


iv) NAND



v) NOR





F = (A.B')+ (C'.D)

ii) Represent Boolean expression using NOR Gates only

F =X(Y'+Z)

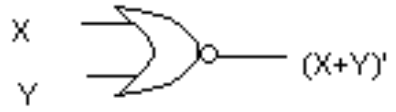
USING INVOLUTION LAW

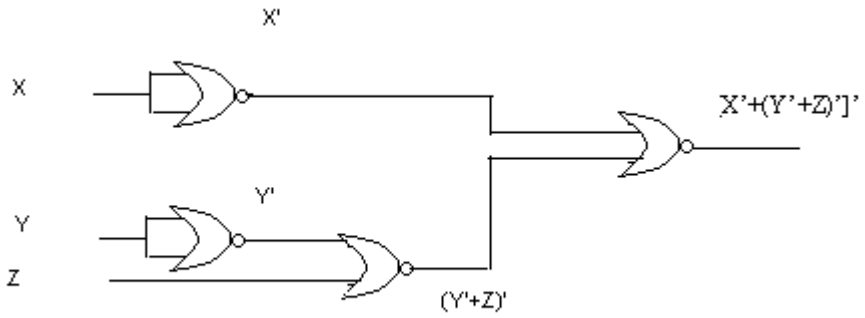
F =F''

X (Y' +Z) = [(X(Y' +Z))']'

BY DE MORGAN'S TH.

[X'+(Y'+Z)']'





6 **(c) Derive SOP/POS of the given Boolean function.**

∑SOP -> sum of Product => e.g. ab+cd

Π POS => Product of Sum => eg. (a+ b) . (c+ d)

If SOP => F (X,Y,Z) = ∑ (1,2, 5,7)

Then its POS => F (X,Y,Z) = Π (0,3, 4,6)

Note :- For 3 variables the combination are 8 i.e.

0 1 2 3 4 5 6 7

000 001 010 011 100 101 110 111

If the combination (any) belongs to SOP then remaining belongs to POS.

e.g. SOP is ∑ (1,2,5,7) then POS will be Π (0,3,4,6) and vice versa ∑ (1,2,5,7) => Π (0,3,4,6)
(x'yz' + x'yz'+ xyz' +xyz) <=> (x+y+z) (x+ y'+z) (x'+ y+ z) (x'+y'+z)

Here 0 will be as complement	**Here 1 will be as complement**
(x'y'z) + (x'yz') + (xy'z) + (xyz)	**(x+y+z) (x+y'+z) + (x'+y+z) + (x'+y'+z)**
001 010 101 111	**000 011 100 110**
1 2 5 7	**0 3 4 6**

Given TT derive its SOP &POS

SNO.	X	Y	Z	F(X,Y,Z)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1

SOP will be

F(XYZ) = ∑ (1,2,5,7) POS will be F(XYZ) = Π (0,3,4,6)

e.g. SOP ∑ (0,1,5,6) ⇔ POS Π (2,3,4,7)

SOP ∑ (4,5,6,7) ⇔ POS Π (0,1,2,3)

6(d) Obtain a simplified form for the following Boolean Expression using **K- Map**

F(u,v,w,z) = ∑ 0,3,4,5,7,11,13,15)

2 Quads (3,7,15,11) & (5,7,13,15)

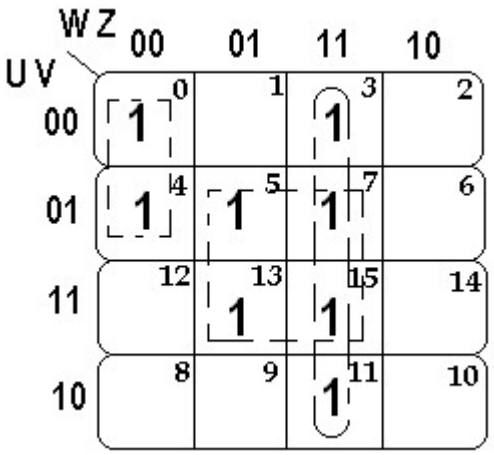
1 Pair (0,4)

Quad1 ($m_3 + m_7 + m_{11} + m_{15}$) = **WZ**

Quad2 ($m_5 + m_7 + m_{13} + m_{15}$) = **VZ**

Pair1 ($m_0 + m_4$) = $\overline{\mathbf{UWZ}}$

F = WZ +VZ+ $\overline{\text{UWZ}}$



Note: ∑ (SOP)=> 0 represents as complement & 1 is marked in K-Map

K- Map

F(u,v,w,z) = □ Π (0,3,4,5,7,11,13,15)

2 Quads (3,7,15,11) & (5,7,13,15)

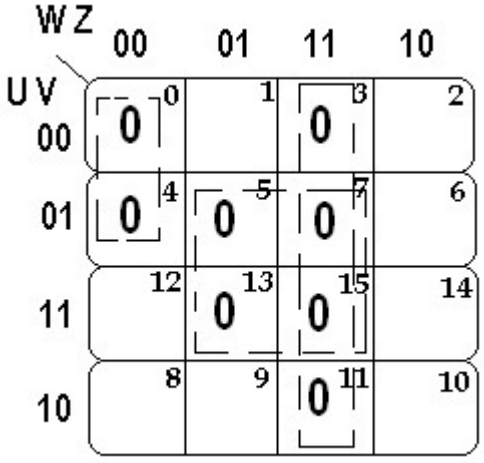
1Pair (0,4)

Quad I ($m_3 . m_7 . m_{11} . m_{15}$) = **W+Z**

Quad 2 ($m_5 . m_7 . m_{13} . m_{15}$) = **V+Z**

Pair 1 ($m_0 + m_4$) = **U+W+Z**

F = **(W+Z) . (V+Z) .(U+W+Z)**



Note Π(POS) => 1 represents as complement & 0 is marked in K-Map

COMPUTER NETWORKING

7(a) **Abbreviations**

Abbreviation	Full Form
LAN	Local Area Network
WAN	Wide Area Network
MAN	Metropolitan Area Network
FTP	File Transfer Protocol
SMTP	Simple Mail Transfer Protocol

IMAP	Internet Message Access Protocol
MODEM	Modulation And Demodulation
WWW	World Wide Web
RPC	Remote Procedure Call
NFS	Network File System
HTML	Hyper Text Markup Language
DHTML	Dynamic Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
TCP/ IP	Transmission Control Protocol Internet Protocol
SLIP	Serial Line Internet Protocol
PPP	Point To Point Protocol
SIM	Subscriber's Identification Module
3G	3rd Generation of Mobile Communication Technology
SMS	Short Message Service
EDGE	Enhanced Data rates for Global Evolution
E-mail	Sending / receiving messages electronically
WLL	Wireless Local Loop
CDMA	Code Division Multiple Access
FRA	Fixed Radio Access
GSM	Global System For Mobile Communication
ARPANET	Advanced Research Project Agency Network
XML	Extensive Markup Language
URL	Uniform Resource Locater
ISP	Internet Service Provider
DNS	Domain Name System
VSNL	Videsh Sanchar Nigam Limited
MTNL	Mahanagar Nigam Telephone Limited
WAIS	Wide Area Information Server

TDM	Time Division Multiplexing
WDM	Wavelength Division Multiplexing
FDM	Frequency Division Multiplexing
GNU	GNU's Not Unix
LAMP	Linux, Apache, MySQL,PHP
WIMP	Windows,IIS,MySQL,PHP
Bps	Bits per second
W3C	World Wide Web Consortium
HTTP	Hyper Text Transfer Protocol
NIU	Network Interface Unit
TAP	Terminal Access Protocol
PHP	Hypertext Preprocessor

REPEATER-Repeater is a network device that amplifies and restores signals for long distance transmission.

BRIDGE- A bridge is a network device that is used to establish connection between two local networks with the same standard local networks with the same standard but with different types of cables.

ROUTER-A Router is a network device that is used to separate different segments or networks and can handle different protocols.

GATEWAY- Network device to connect dissimilar networks.

PROTOCOLS- It is formal description of message format and the rules that two or more machines must follow to exchange those messages.

TELNET- Internet utility used for remote login.

FIREWALL- A Firewall is a system designed to prevent unauthorized access to or from a private network.

COOKIES- Cookies are the messages that a web server transmits to a web server can keep track of the user's activity on a specific website.

CYBER LAW- All legal and regulatory aspects of Internet and WWW are designed by cyber law.

CRACKERS- Malicious Programmers who break into secure systems are called crackers.

HACKERS- Programmers who gain knowledge about computer systems for playful pranks are known as Hackers.

MODEM- Computer peripheral that allows to connect & communicate with other computers via telephone lines. It converts digital data into analog data and vice- versa.

7) **ii) Suitable Place of Server**

The most suitable place to house the server of this organization would be that block which contains the maximum number of computers which decrease the capability cost acc to 80 20 network design rule.

iii) **Very effective (High Speed) Communication- OPTICAL FIBRE**

High Speed internet connectivity- BROAD BAND

High speed connectivity in Hilly Regions- RADIO/MICROWAVES

iv) **HUB is required for LAN to connect multiple computers to the server. REAPEATER is required to amplify signals beyond 50 m distance**.

OSS- Open Source Software : These s/w are freely available for the user including source code. He can modify but it does not have to be free of charge..

FSF- Free Software Foundation:- Free software are freely available but free to be used, copied, studied, modified and redistributed in binary form. No payments are needed to be made for free soft wares.

Proprietary software available at no cost is **Freeware.**

Shareware is available at Zero price for trial period. It includes built in timed mechanism which limits functionality after trial period.

Website:- Location on a net server.

Web pages:- Documents residing on websites.

SNO.	HTML	XML
1	Document layout & hyperlink specification language	Language for documents containing structured information.
2	Tag semantics & tag set are fixed(tags are the layout commands that let us to control the presentation of information on web pages)	It provides facility to define tags and the structural relationship between them.

****