Q. 1.a Ordinary function: These are functions defined anywhere in the program and called directly using function name.

Example

```cpp
void cube(int x)
{
    cout << x * x * x;
}

void main()
{
    int a;
    cin>>a;
    cube(a); //Function call.
}
```

Member function: These are functions defined inside the class and called using the object.

```cpp
class A
{
    int x;

    public:

    void cube()
    {
        cout << x * x * x;
    }
}

void main()
{
    A1 = A1.cube(); //Function call
}
```

(½ Mark for each correct explanation and ½ Mark for each correct example of ordinary and member function)

b. (i) getchar ()

(ii) isalnum ()

(½ Mark for writing each correct library function name)

c. # include < iostream.h >

# include < math.h >
# define PI 3.14
void main()
{
float r, a;
cout << "enter any radius ";
cin >>r;
a = PI * pow (r, 2);
cout << "Area = " << a ;
}
(½ Mark for each correction).

d. Text = tMKCM@IMJGCR
New Text = CM@IMJGCR
last Text = IMJGCR

[ 1 Mark for first line
½ Mark for second line
½ Mark for third line]

e. 5 : B : 55
1 : B : 50
5 : C : 85
[1 Mark for each correct line of output]

f. (iii) O – R – A – G –
Minimum L value – 5
Maximum L value – 8
[1 Mark for correct option ½ Mark for each min and max value of L]

Q. 2. a Encapsulation : Wrapping up of data and associated function into a single group is called encapsulation.

Abstraction : Act of representing essential feature without including background detail is called abstraction.

In C++ encapsulation is implemented by using class and abstraction is implanted by using private access mode.

Ex.
oclass Stu

Abstraction char name[20];

Encapsulation

public:
    void input ( );
    void output ( );

};

[½ mark for each correct defn, and ½ mark for example]

b. Constructor overloading

    void stream :: Steam (int Sc, char S[], float f)
    {
        streamcode = Sc;
        strcpy (streamname, S);
        fees = f;
    }

    [½ Mark for constructor overloading]
    (½ Mark for defn.)

ii) Statement 1 – implicit call
    Statement 2 – Explicit call

Implicit call – It will not create temporary object.
Explicit call – It will create temporary object with class name.

[½ mark for each correct name]

c. class customer

private :
    int customer_no;
    char customer_name [20];
    int Qty;
    float price, Totalprice, Discount, Netprice;

public :
    customer()
    {
        customer_no = 111;
        strcpy (customer_name, “Leena”);
        Qty = 0;
        Price = Totalprice = Netprice = Discount = 0;
    }

    void input ( )
Enter 'customer_no, Customer_name, Qty and Price';
cin>> customer_no;
gets (Customer_name);
cin>> Qty;
cin>> prices;
Caldiscount();
}
void Caldiscount ( );
void show ( )
{
cout << "customer_no :"<<customer_no;
cout << "\n name :"<<customer_name;
cout << "\n prince :"<<price;
cout << "\n Qty :"<< Qty;
cout << "\n Total price :"<<Totalprice;
cout << "\n Discount :"<<Discount;
cout << "\n Net price :"<< Netprice;
}
};
void customer :: Caldiscount( )
{
Totalprice = price * Qty;
if (Totalprice >= 50000)
    Discount = 25 * Totalprice / 100;
else if (Totalprice >= 25000)
    Discount = 15 * Totalprice / 100;
else
    Discount = 10 * Totalprice / 100;
}

(i) Dealer – 118
Accessories – 94
(ii) Multilevel inheritance
Base class – AC
Derived class – Dealer
[½ mark for correct option]
[½ mark for correct base and derived class name]

(iii) Data member

<table>
<thead>
<tr>
<th>Data member</th>
<th>Member function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>enteraccessoriesdetails ( )</td>
</tr>
<tr>
<td></td>
<td>showaccessoriesdetails ( )</td>
</tr>
<tr>
<td></td>
<td>enterdetails ( )</td>
</tr>
<tr>
<td></td>
<td>showdetails ( )</td>
</tr>
</tbody>
</table>

[½ mark for correct Data Members]
[½ mark for correct member functions]

(iv) Data member

<table>
<thead>
<tr>
<th>Data member</th>
<th>Member function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>enterdetails ( )</td>
</tr>
<tr>
<td>No_of_dealers</td>
<td>showdetails ( )</td>
</tr>
<tr>
<td>Dealers_name</td>
<td>enteraccessoriesdetails ( )</td>
</tr>
<tr>
<td>No_of_products</td>
<td>showaccessoriesdetails ( )</td>
</tr>
<tr>
<td>Stabilizer</td>
<td>entercarddetails ( )</td>
</tr>
<tr>
<td>AC_Cover</td>
<td>voidshowcarddetails ( )</td>
</tr>
</tbody>
</table>

(½ Mark for correct Data members)
(½ Mark for correct member functions)

Q.3. a)

\[ m = 37 \]
\[ n = 18 \]
\[ w = 4 \text{ bytes} \]
\[ I_o = -1 \]
\[ J_o = -2 \]
\[ T[1][J] = B + W[(I - I_o)n + (J - J_o)] \]
\[ T[2][2] = 3000 \]
\[ T[20][5] = ? \]
\[ T[2][2] = B + 4[2 - (-1)x18 + (2 - (-2))] \]
\[ 3000 = B + 4[3x18 + 4] \]
\[ = B + 4[54 + 4] \]
\[ = B + 4[58] \]
\[ 3000 = B + 232 \]
\[ B = 3000 - 232 = 2768 \]

\[ T[20][5] = 2768 + 4 \left[(20 - (-1)) \times 18 + (5 - (-2))\right] \]
\[ = 2768 + 4 \left[21 \times 18 + 7\right] \]
\[ = 2768 + 4 \left[378 + 7\right] \]
\[ = 2768 + 4 \times 385 \]
\[ = 2768 + 1540 \]
\[ = 4308 \]

Total number of elements = 37 \times 18 = 666
Total bytes = 4 \times 666 = 2664 \text{ bytes}

[b) void SORTSCORE (IPL I[], int n )
{ int i, POS, j; }
IPL small, temp;
for ( i = 0; i < n-1; i + + )
{ POS = i;
 small = I[i];
 for ( j = i + 1; j < n; j + + )
 { }
 if ( I[j].score > small.score)
 { POS = j;
  Small = I[j];
 }
 temp = I[i];
 I[i] = I[POS];
 I[POS] = temp;
}
}]

\[ \frac{1}{2} \text{ mark for correct function Header} \]
c) struct Game
{ char Game[30];
int numberofplayer;
Game * next;
};
class stack
{
Game * temp, * top;
public :
    stack ( )
    { top = NULL; 
    }
    void POP ( ) ;
    void push ( ) ;
};
void stack :: POP ( )
{                  
    if ( top = = NULL)
        cout << “Stack empty”;
    else
        { temp = top;
            top = top- >next;
            delete (top);
        }
}
void stack :: push ( )
{
    temp = new (Game);
    cin >> temp->Game[30];
    cin >> temp->numberofplayer;
    temp->next = top;
    top = temp;
}
( ½ mark for class defn.)
( ½ mark for constructor with top = NULL)
d) ```
void sumnegative (int A[10][10], int n)
{
    int s = 0, i, j;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            if ((i == j) && (A[i][j] < 0))
                s += A[i][j];
            if ((i+j == n-1) && (A[i][j] < 0))
                s += A[i][j];
        }
    }
cout << " Total = " <<s;
}
```

(½ mark for push function)
1 ½ mark for POP function
(½ mark for for loop)
(½ mark for checking i == j, i + j == n - 1 and A[i][j]<0)
(½ mark for finding sum)
(½ mark for display sum)

e) 2, 13, +, 5, -, 6, 3, /, 5, *, <

<table>
<thead>
<tr>
<th>STACK</th>
<th>OPERATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2, 13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>+</td>
</tr>
<tr>
<td>15, 5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>10, 6</td>
<td></td>
</tr>
<tr>
<td>10, 6, 3</td>
<td>/</td>
</tr>
<tr>
<td>10, 2, 5</td>
<td>*</td>
</tr>
<tr>
<td>10, 10</td>
<td>&lt;</td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Output 0 (False)

[½ mark for finding result upto '+' operator
½ mark for finding result upto '-' operator]
Q. 4. a)  
```cpp
file . seekp ( 9 * size of (STUDENT), ios : : beg) ;
file . seekp (3 * size of (STUDENT), ios : : beg);
```

(½ mark for each correct statements)

b)  
```cpp
void counthree ( )
{
    ifstream infile ( " VOWEL . TXT");
    char c [ 20 ]; int count = 0;
    if ( ! infile )
        cout << "Not exist";
    else
    {
        infile . getline ( c, 20, ' ' );
        if (strlen (c ) == 3)
            count ++ ;
    }
    cout << "Total count =" << count;
}
```

(½ mark for opening file)
(½ mark for reading string)
(½ mark checking and counting number of 3 character words)
(½ mark for printing count)

c)  
```cpp
void fun ( )
{
    CAR C;
    ifstream infile ( " CAR.DAT", ios : : binary) ;
    if ( ! infile )
        cout < < " not exit ";
    else
    {
        while (infile . read (( char * ) &C, size of(C )))

        {
            if ( C. RETURN_Milage () >= 100 && C. RETURN_Milage( ) <= 150)
                C. display ( );
        }
```
(1 mark for opening CAR.DAT correctly)
(½ mark for reading records)
(½ mark for comparing millage)
(1 mark for displaying record)

SECTION – B

Q.1 a)

<table>
<thead>
<tr>
<th>Static Method</th>
<th>Instance Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>i) are declared with keyword static</td>
<td>No keyword required</td>
</tr>
<tr>
<td>ii) self is not required as first argument</td>
<td>first argument to function is &quot;self&quot;</td>
</tr>
<tr>
<td>iii) invoked using class qualifier</td>
<td>invoked using method qualifier</td>
</tr>
<tr>
<td>iv) faster in execution</td>
<td>slower in execution</td>
</tr>
<tr>
<td>v) example</td>
<td>Example</td>
</tr>
</tbody>
</table>

(½ mark each for any 2 correct points of comparison)

b) i) search(), find(), findall() of string module.

(½ mark for any correct method name)
(No marks to be deducted for not mentioning module name)
ii) index ( )

(½ mark for the correct function name)

c) import math OR from math import pow

def main ( ) :
    r = float (raw_input ( 'enter any radius : '))
    a = 3.14 * math.pow (r, 2) OR a = 3:14 * pow (r, 2)
    print "Area = ", a

(½ mark for each correction done. ½ * 4 = 2 for any 4 corrections. Similar type of correction are to be considered one).

(deduct ½ mark for not underlining the corrections)

(Any other correct solution should be considered)

d) 3

The expression will be x = x + x – x, substituting value of x as 3 we get 3 + 3 – 3 which is 3.

(1 mark for correct answer)

(1 mark for correct justification)

e) 52

Age is assigned value 50, and dictionary contains two elements id & age, so sum is 52.
(1 mark for correct answer)
(2 mark for correct justification)

f) (iii) option , max value = 8, min value 5,

(1 mark for correct option number or option answer)
(1 mark for correct justification)

Q.2. a) Encapsulation : Wrapping up of data and associated function into a single group is called encapsulation.

Abstraction : Act of representing essential feature without including background detail is called abstraction.

Example

class stu (object) :                # data abstraction
    
    def __init__(self):
        self.rollno = 0               # indentation is used for encapsulation
        self.name= ' '
        self.fees = 0.0

    def input (self) :
        self.rollno = input ( )
        self.name = raw_input ( )
        self.fees   = input ( )

    def output (self) :
        print self.rollno
        print self.name
print self.fees

(½ mark each for correct definition)

(½ mark each for specification of abstraction and encapsulation in example)

b)

A
B
F

A is ‘print A’ lies outside try……….except statement it will always be executed.

B is printed as it is first statement of try

Calculation of ‘a’ results in run time error (division by 0 ) so clause ZeroDivisionError is executed, hence value F is printed

(½ mark for first two statement)
(½ mark for last statement)
(1 mark for justification)

c) class customer(object):

def __init__(self):
    
    self.customernumber = 111

    self.customername = ‘Leena’

    self.qty = 0

    self.price, self.discount, self.netprice = 0, 0, 0

def caldiscount(self):

    self.totalprice = self.price * self.qty

    if self.totalprice >= 50000:
self.discount = 0.25

elif 25000 >= self.totalprice < 50000:
    self.discount = 0.15

else:
    self.discount = 0.10

self.netprice = self.totalprice * self.discount

def input(self):
    self.customername = raw_input('enter customer name :')
    self.customernumber = input('enter customer no.)
    self.qty = input('enter quantity')
    caldiscount()

def show(self):
    print "customer number is :", self.customernumber
    print "customer name is :", self.customername
    print "quality purchased :", self.qty
    print "price of item :", self.price
    print "net price is :", self.netprice

(1 mark for correct syntax of class)

(½ mark for correct definition of init function)

(½ mark for correct definition of show ( ) )

(1 mark for correct definition of input ( ) with proper invocation of caldiscount())

(1 mark for correct definition of caldiscount())

d) Example
class Base(object):
    def __init__(self):
        self.a = 0
        self.b = 0

class derived(Base):
    def __init__(self):
        super(Base, self).__init__(self)
        self.c = 0

Example

class Base(object):
    def __init__(self):
        self.a = 0
        self.b = 0

class derived(Base):
    def __init__(self):
        Base.__init__(self)
        self.c = 0

(1 mark for either explanation / usage the super ( ))

(1 mark for proper explanation / usage of Base class method with base class id)

e) import math
def lsum(list):
    sum = 0
    try:
for val in list:
    sum += math.sqrt(val)

except ImportError:
    print "you forgot to import math module"

except TypeError:
    print "list contains non numeric values"

finally:
    return sum

(½ mark for correct function header)

(½ mark each for correctly checking Import Error & Type Error)

(½ mark for return of sum, in any way)

Q.3.a) **Bubble sort**

1\textsuperscript{st} pass :- 40, -23, 11, 27, 38, -1, 67

2\textsuperscript{nd} pass :- -23, 11, 27, 38, -1, 40, 67

3\textsuperscript{rd} pass :- -23, 11, 27, -1, 38, 40, 67

**Selection Sort**

1\textsuperscript{st} pass :- -23, 67, 40, 11 27, 38, -1

2\textsuperscript{nd} pass :- -23, -1, 40, 11, 27, 38, 67

3\textsuperscript{rd} pass :- -23, -1, 11, 40, 27, 38, 67

(½ mark for each correct pass)

b) def bsearch (list, val):
lb = 0
ub = len(list) - 1

while lb <= ub:
    mid = (lb + ub) / 2
    if list[mid] == val:
        return 1
    elif list[mid] < val:
        lb = mid + 1
    else:
        ub = mid - 1

return 0

(½ mark for definition of lb & ub )
(½ mark for correct re-definition of lb & ub )
(½ mark for correct loop )
(½ mark for returning correct value )
(any alternative code giving correct answer is acceptable)
d)

```python
import math

def is_prime(numb):
    if numb > 1:
        if numb == 2:
            return True
        if numb % 2 == 0:
            return False
        for i in range(3, int(math.sqrt(numb)) + 1, 2):
            if numb % i == 0:
                return False
        return False

def get_primes(num):
    while True:
        if is_prime(num):
            yield num
        num += 1
```

(½ mark for checking constant 1 & 2)
(½ mark for eliminating even numbers)
(1 mark for checking of an odd number)
(1 mark for correctly using yield statement)

e) \[ \begin{align*}
2, 13, +, 5, -, 6, 3, /, 5, *, <
\end{align*} \]

<table>
<thead>
<tr>
<th>STACK</th>
<th>OPERATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2, 13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>+</td>
</tr>
<tr>
<td>15, 5</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>-</td>
</tr>
<tr>
<td>10, 6</td>
<td></td>
</tr>
<tr>
<td>10, 6, 3</td>
<td>/</td>
</tr>
<tr>
<td>10, 2, 5</td>
<td>*</td>
</tr>
<tr>
<td>10, 10</td>
<td>&lt;</td>
</tr>
</tbody>
</table>

Output: 0 or False

(½ mark for finding result upto ‘+’ operator)
(½ mark for finding result upto ‘-’ operator)
(½ mark for finding result upto ‘/’ operator)
(½ mark for finding result upto ‘<’ operator)
(2 marks for correct answer with proper step)
(½ mark for only correct answer)

Q.4.a) write ( ) is used to write a string in the text file
writelines ( ) is used to write a string, list, tuple or dictionary in the data file.
b) def carfile():
    ifile = open ('car.txt', 'r')
    line = ifile.readline()
    while line :
        x = line.split ( )
        if 100 >= x[2] <= 150 :
            print line
        line = ifile.readline()
    ifile.close()

(½ mark for reading a line from the file)

(½ mark for loop)

(½ mark for splitting the content of string read from the file)

(½ mark for correct comparison & printing)

c) def fileprocess ( ) :  
    import pickle

    list = [ ]
    sum = 0
    count = 0

    file = open ('log.dat', 'rb')
    List = pickle.load (file)
    for i in List :
x = i.split()
if x[0].find('xerrox') == 0 :
    y = float(x [1])
    sum +  = y
    count +  = 1
    avg = sum / count
    print avg,  count

(½ mark for loading the file)

(½ mark for the loop)

(½ mark for comparsion)

(½ each for sum and avg calculation)

(½ mark for print)

SECTION – C

Q.5. a) Degree —— number of columns in a table
Cardinality —— number of rows in a table.

Degree — 4
Cardinality — 5
(½ mark each for correct concept of degree and cardinality)
(½ mark each for correct degree and cardinality value)

b) (i) Select comname
    from company, customer
    where company. CID = customer . CID
    and price < 30000;

[½ mark for First two line and ½ mark for where clause ]
(ii) select name 
    from company 
    order by name desc;

[½ mark for select and from and ½ mark for order by command]

(iii) update customer 
    set price = price + 1000 
    where name like ' S% ' ;

[½ mark for update and set and ½ mark for where clause]

(iv) Alter table customer 
    Add totalprice decimal ( 10, 2 );

[½ mark for First line and ½ mark for second line ]

(v)

<table>
<thead>
<tr>
<th>Count(*)</th>
<th>CITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>DELHI</td>
</tr>
<tr>
<td>2</td>
<td>MUMBAI</td>
</tr>
<tr>
<td>1</td>
<td>MADRAS</td>
</tr>
</tbody>
</table>

[½ mark for correct answer ]

(vi) MIN (PRICE)                MAX (PRICE)
    50000              70000

[½ mark for correct answer ]

(vii)

<table>
<thead>
<tr>
<th>AVG (QTY)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

[½ mark for correct answer]

(viii) | PRODUCT NAME | CITY | PRICE |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MOBILE</td>
<td>MUMBAI</td>
<td>70000</td>
<td></td>
</tr>
</tbody>
</table>
Q. 6 a) Principle of duality states that any theorem/identity/statement in Boolean algebra remains true if identity element \((0,1)\) and operators \((+, .)\) are swapped.

Importance – It is sufficient to prove one part of them/identity/statement).

(1 mark for correct definition or an example depicting the correct definition)
(1 mark for its importance represented in any way)

b) 

\[
(x + y')'.(x'+y)
\]

OR

\[
(x'y).(xy')
\]

(½ mark each for correct expansion for NAND gate)
(1 mark for the term of NOR gate)

c) 

\[
\Pi (0, 1, 4, 7, 8, 9, 12, 13, 14)
\]

OR

\[
M0 \ M1 \ M4 \ M7 \ M8 \ M9 \ M12 \ M13 \ M14
\]

OR

\[(a+b+c+d). (a+b'+c+d') . (a+b'+c+d) . (a'+b+c+d) . (a'+b+c+d') \]

\[(a'+b'+c+d'). (a'+b'+c+d') . (a'+b'+c+d)
\]

(1 mark for correct answer)
(½ for mentioning correct five term)
Q.7. a) **Optical Fibres**

   (i) It is immune to electrical and magnetic interference
   (ii) It is highly suitable for harsh industries environments
   (iii) Very high transmission capacity
   (iv) Secure transmission
   (v) It is used for broadband transmission.

   [Any two option - 1 mark]
b)  

b 1)  

[ 1 mark for correct layout ]

b 2)  
ADMIN – because more number of computers  
[ ½ mark for suitable place and ½ mark for suitable reason ]

b 3)  
Repeater → admin to senior  
Hub / Switch → all building  
[ ½ mark for repeater and ½ mark for switch / Hub ]

b 4)  
Radio wave.  
[ 1 mark for correct option ]

c)  
URL → http://www.income.in / home.aboutus.html  
Domain name → www.income.in  
[1/2 mark for URL and ½ mark for Domain name]

d)  
Web Hosting: web hosting is the process of uploading / savig the web content on a web server to make it available on www.  
[ 1 mark for correct answer ]

e)  
Circuit Switching  
Packet Switching
- Complete physical connection - No physical connection
  is established between nodes is established
- No fixed size - Fixed Size

[ Any two correct difference 1 mark (i.e.) ½ mark each]

e) Telnet.
[ 1 mark for correct protocol ]

f) Firewall is hardware or software based network security system. It prevents
unauthorized access to or from a network.
[ 1 mark for proper defn.]