DEFINITION:

1. GET A NAME

2. TWO NUMBERS

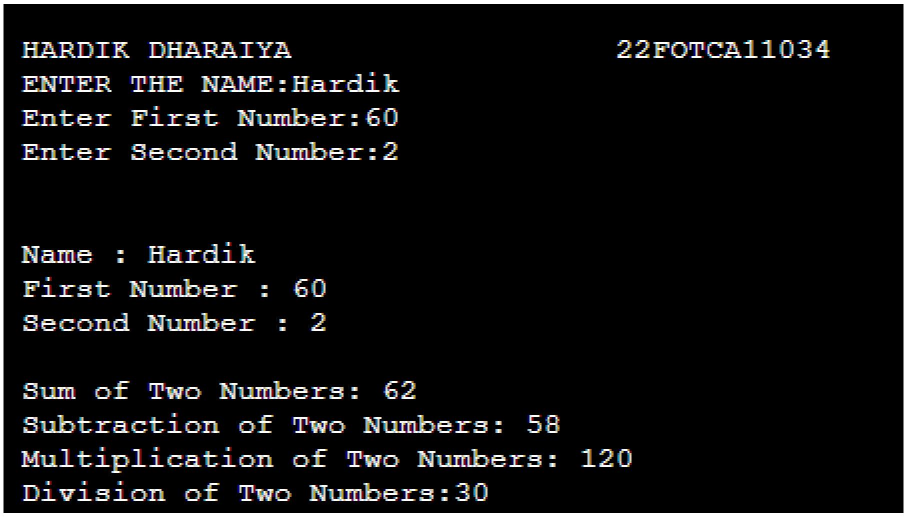
FROM THE USER DO

3. SUM OF TWO NUMBERS.

**C
O
D
E**

```
#include<iostream>
using namespace std;
int main()
{
   char name[20];
   int num1, num2;

   cout <<"\n HARDIK DHARAIYA              22FOTCA11034 \n ENTER THE NAME:";
   cin >>name;

   cout <<" Enter First Number:";
   cin >>num1;

   cout <<" Enter Second Number:";
   cin>>num2;


   cout<<"\n\n Name : "<<name;
   cout<<"\n First Number : "<< num1;
   cout<<"\n Second Number : "<< num2;

   cout<<"\n\n Sum of Two Numbers: " <<num1+num2;
   cout<< "\n Subtraction of Two Numbers: " << num1-num2;
   cout<<"\n Multiplication of Two Numbers: " << num1*num2;
   cout<<"\n Division of Two Numbers:" << num1/num2;

   return 0;
}
```

SCREEN SHOT



```
HARDIK DHARAIYA                         22FOTCA11034
ENTER THE NAME:Hardik
Enter First Number:60
Enter Second Number:2


Name : Hardik
First Number : 60
Second Number : 2

Sum of Two Numbers: 62
Subtraction of Two Numbers: 58
Multiplication of Two Numbers: 120
Division of Two Numbers:30
```

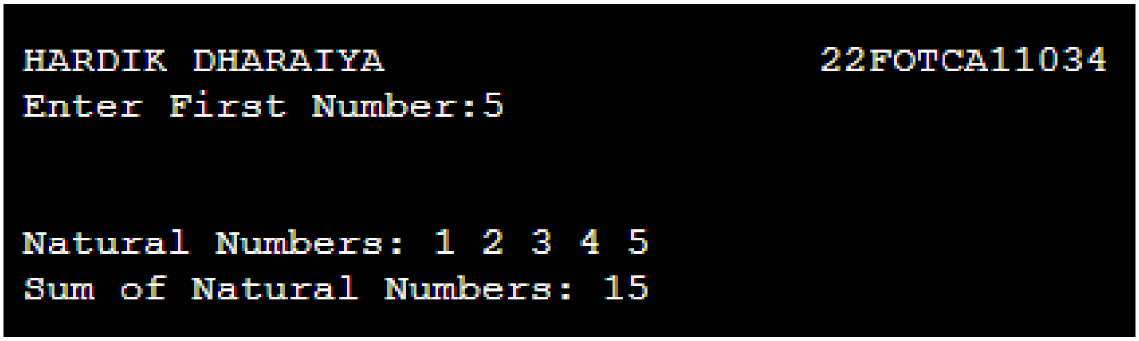
4. DEFINITION:

SUM OF NATURAL NUMBERS.

CODE

```cpp
#include<iostream>
using namespace std;
int main()
{
    int h, num, sum=0;

    cout <<"\n HARDIK DHARAIYA                  22FOTCA11034";

    cout <<"\n Enter First Number:";
    cin >>num;

  cout<<"\n\n Natural Numbers: ";
  for (h=1; h<=num; h++)
  {
      cout<<h<<" ";
      sum = sum+h;
  }

  cout<<"\n Sum of Natural Numbers: "<< sum;

    return 0;
}
```

SCREEN SHOT



```
HARDIK DHARAIYA                  22FOTCA11034
Enter First Number:5


Natural Numbers: 1 2 3 4 5
Sum of Natural Numbers: 15
```

5.

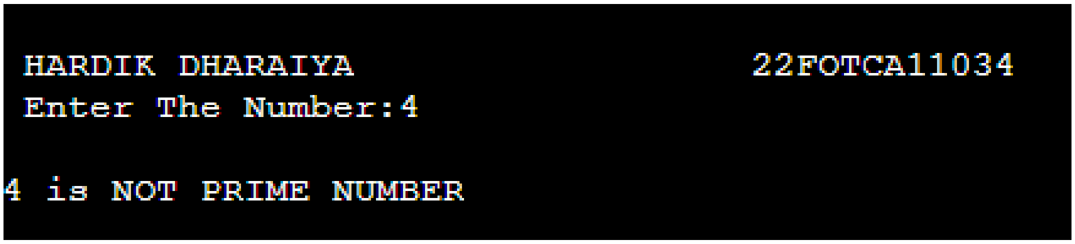
DEFINITION:

WAP TO CHECK WHETHER THE NUMBER IS PRIME OR NOT.

C O D E

```
#include<iostream>
using namespace std;
int main()
{
    int h, num, flag = 0;

    cout <<"\n HARDIK DHARAIYA                  22FOTCA11034";

    cout <<"\n Enter The Number:";
    cin >>num;

    if(num==0 || num==1)
    {
        flag=2;

    }
    else if(num>=2)
    {
        for (h=2; h<=num/2; h++)
        {
            if ((num % h) == 0)
            {
                flag=1;

            }

        }
    }

    if (flag==0)
    {
        cout<<"\n"<<num<<" is PRIME NUMBER";
    }
    else
    {
        cout << "\n"<< num<<" is NOT PRIME NUMBER";
    }

    return 0;
}
```

OUTPUT



```
HARDIK DHARAIYA                    22FOTCA11034
Enter The Number:4

4 is NOT PRIME NUMBER
```

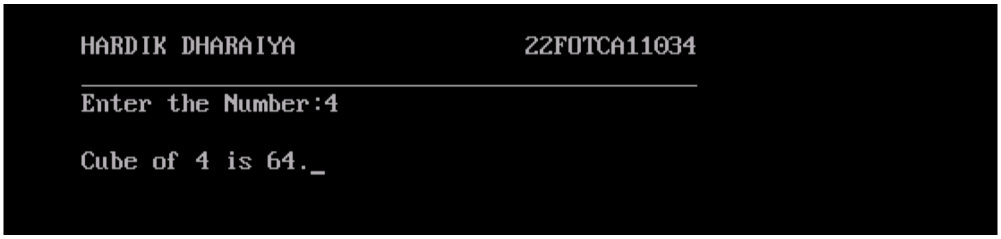
6. **DEFINITION:**

WAP TO PRINT CUBE OF NUMBER. (TURBO C++)

CODE

```
#include<iostream.h>
#include<conio.h>
void main()
{
        int h=1, cube=1, num=0;

        clrscr();


        cout<<"\n HARDIK DHARAIYA                22FOTCA11034";
        cout<<"\n ____________________________________________";
        cout<<"\n Enter the Number:";
        cin>>num;

        cube=num*num*num;

        cout<<"\n Cube of "<<num<<" is "<<cube<<".";

        getch();
}
```

OUTPUT



```
HARDIK DHARAIYA                22FOTCA11034
_____________________________________________
Enter the Number:4

Cube of 4 is 64._
```

7.

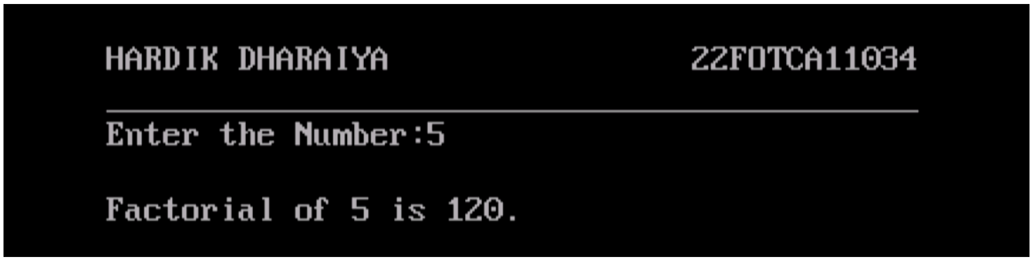
DEFINITION:

WAP TO PRINT FACTORIAL OF NUMBER. (TURBO C++)

C O D E

```
#include<iostream.h>
#include<conio.h>
void main()
{
        int h=1, facto=1, num=0;

        clrscr();


        cout<<"\n HARDIK DHARAIYA                    22FOTCA11034";
        cout<<"\n ______________________________________________________";
        cout<<"\n Enter the Number:";
        cin>>num;

        while(h<=num)
        {
                facto=facto*h;
                h++;
        }
        cout<<"\n Factorial of "<<num<<" is "<<facto<<".";

        getch();
}
```

O U T P U T



```
HARDIK DHARAIYA                    22FOTCA11034
______________________________________________
Enter the Number:5

Factorial of 5 is 120.
```

8.

DEFINITION:

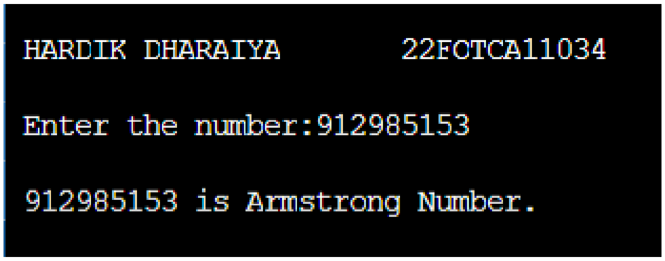
WAP TO CHECK NUMBER IS ARMSTRONG OR NOT.

CODE

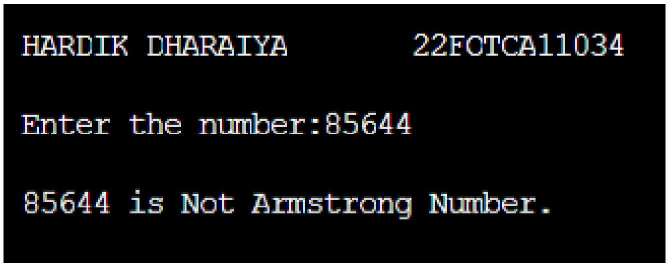
```
#include <iostream>
#include <math.h>

using namespace std;

int main()
{
    long long int num1, rev=0, rem=0, h=0, temp=0, power=0, sum=0, r=1, nu=0;

    cout<<"\n HARDIK DHARAIYA       22FOTCA11034";
    cout<<"\n\n Enter the number:";
    cin>>num1;

    nu=num1;

    //to find total digits in number to give power to each of the digits
    while(nu!=0)
    {
        nu=nu/10;
        power++;

    }


    temp=num1;
    while(num1!=0)
    {
        rem=num1%10; // find a last digit
        sum=pow(rem,power)+sum; // give a power to last digit and add the sum value
        num1=num1/10; // remove last digit from number
    }

    if(temp==sum)
    {
        cout<<"\n "<<temp<<" is Armstrong Number.";
    }
    else
    {
```

```
    cout<<"\n "<<temp<<" is Not Armstrong Number.";
    }
    return 0;
}
```

O
U
T
P
U
T



```
HARDIK DHARAIYA        22FOTCA11034

Enter the number:912985153

912985153 is Armstrong Number.
```



```
HARDIK DHARAIYA        22FOTCA11034

Enter the number:85644

85644 is Not Armstrong Number.
```

9.

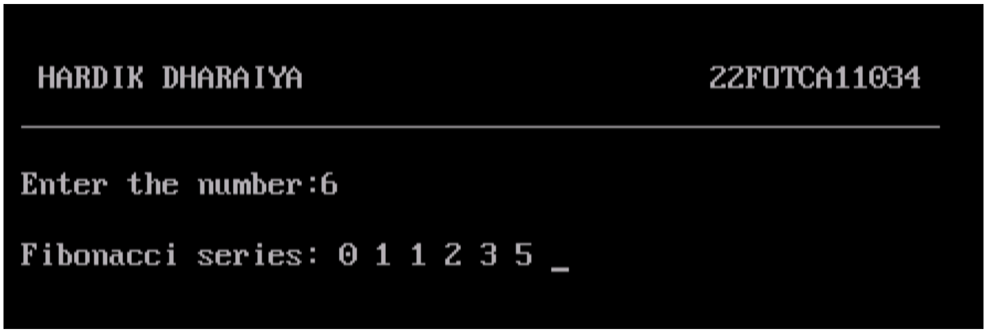
DEFINITION:

WAP TO GENERATE FIBONACCI SERIES. (TURBO C++)

C O D E

```
#include<iostream.h>
#include<conio.h>
void main()
{
    int h=0, num=0, fibo=0, t1=0, t2=1;
    clrscr();
    cout<<"\n  HARDIK DHARAIYA                        22FOTCA11034";
    cout<<"\n __________________________________________________________";
    cout<<"\n\n Enter the number:";
    cin >> num;

    if(num==0)
    {
        cout<<"\n Zero is Not allowed, please enter \n the another number.";
    }
    else if(num==1)
    {
        cout<<"\n Fibonacci series: 0 ";
    }
    else if(num==2)
    {
        cout<<"\n Fibonacci series: 0 1 ";
    }
    else if(num>2)
    {
        cout<<"\n Fibonacci series: 0 1 ";

        for(h=3;h<=num;h++)
        {
            fibo=t1+t2;
            cout<<fibo<<" ";
            t1=t2;
            t2=fibo;
        }
    }

    getch();

}
```

OUTPUT



```
 HARDIK DHARAIYA                        22FOTCA11034
______________________________________________________

Enter the number:6

Fibonacci series: 0 1 1 2 3 5 _
```

10. **DEFINITION:**

WAP TO . (TURBO C++)

CODE

```
#include <iostream>

using namespace std;

int main()
{   int h, r, num, j=1, k, sub;
    cout << "\n\n HARDIK DHARAIYA         22FOTCA11034";
    cout << "\n ------------------------------------";

    cout<<"\n Enter the number:";
    cin>>num;
    sub=num-1;
    h=num;
    while(h>=1)
    {   r=1;
        while(r<=h)
        {
            cout<<" "<<" ";
            r++;
        }
        j=num;
        while(j>=h)
        {
            while(sub>=0)
            {
                k=j-sub;
                cout<<" "<<k;
                sub--;
            }cout<<"\n";
            j--;
        }
        cout<<"\n";
        h--;
    }

    return 0;
}


#include <iostream>

using namespace std;

int main()
{   int h, r, num, j=1, k, sub;
    cout << "\n\n HARDIK DHARAIYA         22FOTCA11034";
    cout << "\n ------------------------------------";

    cout<<"\n Enter the number:";
    cin>>num;
    sub=num-1;
    h=num;
    while(h>=1)
    {   r=1;
        while(r<=h)
        {
            cout<<" "<<" ";
            r++;
        }
        j=num;
        while(j>=h)
        {
            while(sub>=0)
            {
                k=j-sub;
                cout<<" "<<k;
                sub--;
            }
            j--;
        }
        cout<<"\n";
        h--;
    }
    return 0;
}
```

OUTPUT

11.

DEFINITION:

Monal is having a music night function at his home. He told his servant Rajoo to arrange chairs for the guests or participants. Rajoo will arrange chairs rows and columns wise, every single raw number of 6 columns and rows is equal to the number of columns. According to instructions, create a pyramid program using a star pattern.

CODE

```
#include <iostream>

using namespace std;

int main()
{
    int h, r, num, j, k;
    cout << "\n\n HARDIK DHARAIYA         22FOTCA11034";
    cout << "\n ------------------------------------";

    cout<<"\n Enter the number:";
    cin>>num;
    cout<<"\n";

     h=num-1;

    while(h>=1)
    {  r=1;
        while(r<=h)
        {
            cout<<" "<<"+";
            r++;
        }
        j=num-1;
        while(j>=h)
        {
            cout<<" "<<"*";
            j--;
        }

        cout<<"\n";
        h--;
    }


    return 0;
}
```

SCREEN SHOT

12. **DEFINITION:**

WAP to create a staircase for the first floor of the building, the first architect will ask the owner how much length of staircase which he or she wants to implement. Staircase must decrease its length by increasing every number of stairs.

COD

E	
SCREEN SHOT	

13.

DEFINITION:

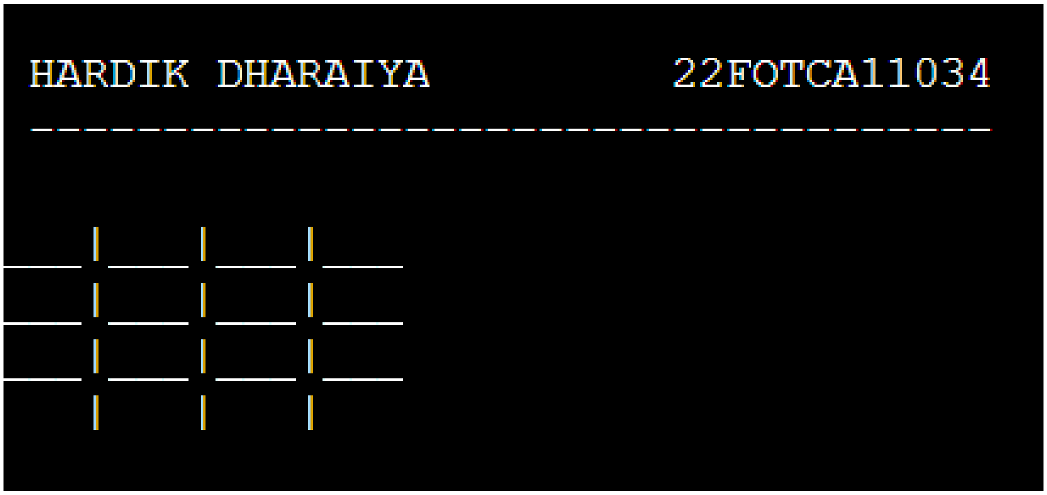
WAP to create a tic tac toe

CODE

```
#include <iostream>

using namespace std;

int main()
{
    int h=1, r=1, num, j, k=0;
    cout << "\n\n HARDIK DHARAIYA         22FOTCA11034";
    cout << "\n ------------------------------------\n\n";

    for(k=1;k<=3;k++)
    {   h=1; r=1;
            num=4;
            while(h<=num)
            {
                while(r<=h)
                {
                    cout<<"___";
                    r++;
                }
                if(h==num)
                {
                    cout<<" ";
                }
                else
                {
                    cout<<"|";
                }
                h++;
            }
        cout<<"\n";
    }
    h=1; r=1;
            num=4;
            while(h<=num)
            {
                while(r<=h)
                {
                    cout<<"   ";
                    r++;
                }
                if(h==num)
                {
                    cout<<" ";
                }
                else
                {
                    cout<<"|";
                }
                h++;
            }

    return 0;
}
```

S
C
R
E
E
N

S
H
O
T



```
HARDIK DHARAIYA          22FOTCA11034
-------------------------------------


_____|_____|_____|_____
_____|_____|_____|_____
_____|_____|_____|_____
     |     |     |
```

14.

DEFINITION:

WAP of type casting or conversion .

C

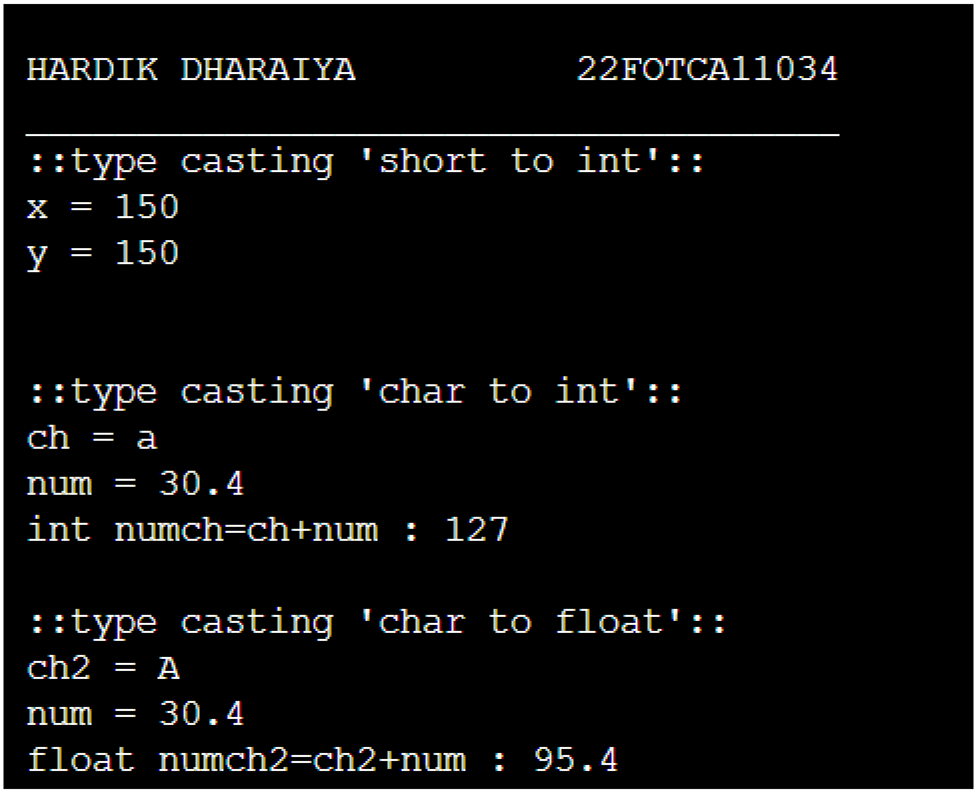
O

D

E

```
#include <iostream>

using namespace std;

int main()
{
   short x=150;
   int y=x;

   float num = 30.4;
   char ch = 'a';
   char ch2 = 'A';

   int numCh=ch+num;

   float numCh2=ch2+num;

   cout<<" \n HARDIK DHARAIYA          22FOTCA11034";
   cout<<" \n _______________________________________";
   cout<<" \n ::type casting 'short to int':: \n x = "<<x<<" \n y = "<<y<<" \n ";

   cout<<" \n\n ::type casting 'char to int':: \n ch = "<<ch<<" \n num = "<<num<<" \n int
numch=ch+num : "<<numCh;

   cout<<" \n\n ::type casting 'char to float':: \n ch2 = "<<ch2<<" \n num = "<<num<<"
\n float numch2=ch2+num : "<<numCh2;

    return 0;
}
```

SCREEN SHOT



```
HARDIK DHARAIYA          22FOTCA11034
_____________________________________
::type casting 'short to int'::
x = 150
y = 150


::type casting 'char to int'::
ch = a
num = 30.4
int numch=ch+num : 127

::type casting 'char to float'::
ch2 = A
num = 30.4
float numch2=ch2+num : 95.4
```

15. DEFINITION:

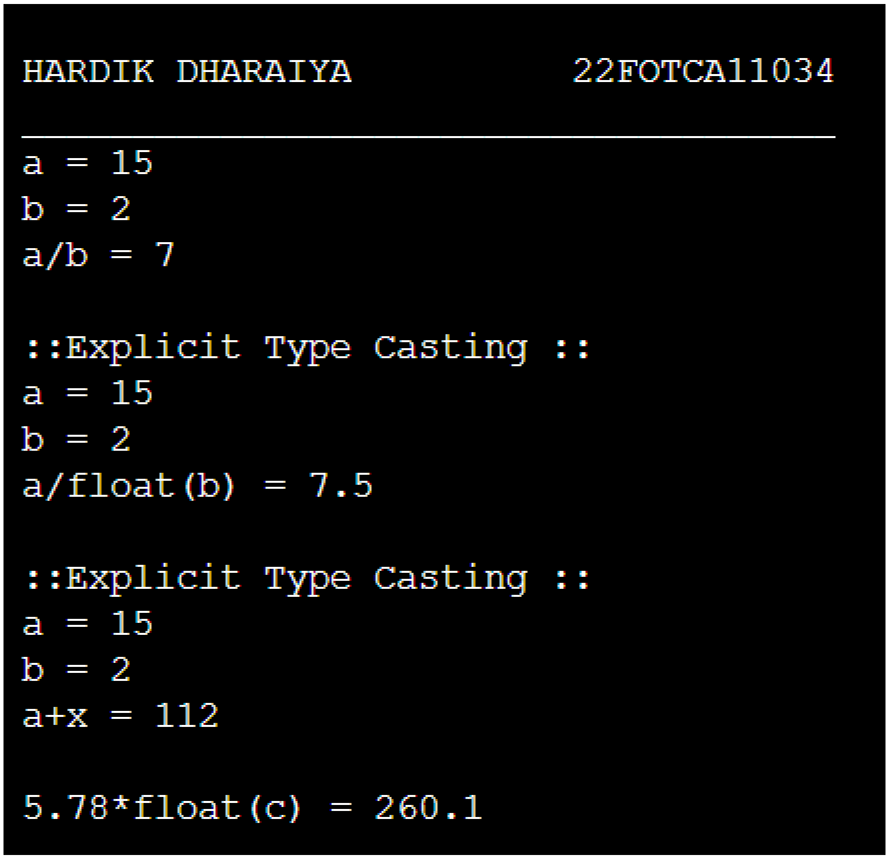
WAP of explicit type casting or conversion .

C
O
D
E

```
#include <iostream>

using namespace std;

int main()
{
    int a, b, c=45.55;
    a=15;
    b=2;

    char ch = 'a';



    cout<<" \n HARDIK DHARAIYA          22FOTCA11034";
    cout<<" \n ________________________________________";
    cout<<" \n a = "<<a<<" \n b = "<<b<<" \n a/b = "<<a/b<<" \n ";

    cout<<" \n ::Explicit Type Casting :: \n a = "<<a<<" \n b = "<<b<<" \n a/float(b) = "<<a/float(b)<<" \n "; //Explicit

    cout<<" \n ::Explicit Type Casting :: \n a = "<<a<<" \n b = "<<b<<" \n a+x = "<<a+ch<<" \n ";

    cout<<" \n 5.78*float(c) = "<<5.78*float(c);

    return 0;
}
```

S
C
R
E
E
N

S
H
O
T



```
HARDIK DHARAIYA          22FOTCA11034
_____________________________________
a = 15
b = 2
a/b = 7

::Explicit Type Casting ::
a = 15
b = 2
a/float(b) = 7.5

::Explicit Type Casting ::
a = 15
b = 2
a+x = 112

5.78*float(c) = 260.1
```

16.

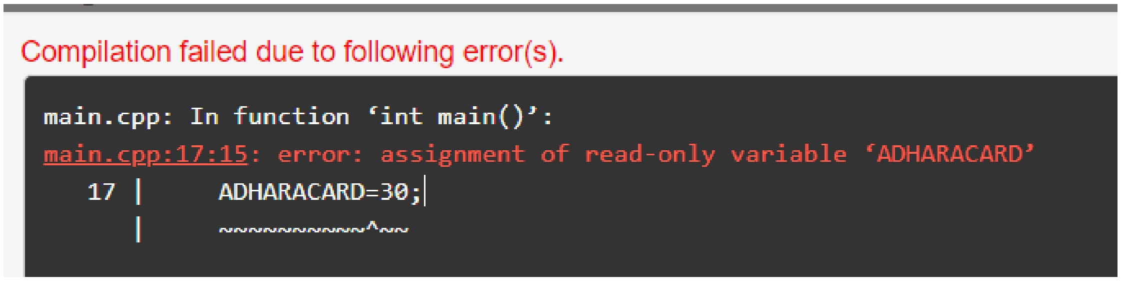
DEFINITION:

WAP of CONST variable .

CODE

```
#include <iostream>

using namespace std;

int main()
{
    const long int ADHARACARD=6565556565;
    ADHARACARD=30;


    cout<<" \n HARDIK DHARAIYA         22FOTCA11034";
    cout<<" \n ________________________________________";

    cout<<" \n ADHARACARD = "<<ADHARACARD;

    return 0;
}
```

SCREEN SHOT



Compilation failed due to following error(s).

```
main.cpp: In function ‘int main()’:
main.cpp:17:15: error: assignment of read-only variable ‘ADHARACARD’
   17 |     ADHARACARD=30;
      |     ~~~~~~~~~~^~~
```

17.	DEFINITION: WAP of CONST variable .
C O D E	
SCREEN SHOT	

18.

DEFINITION:

LADDER class.

C

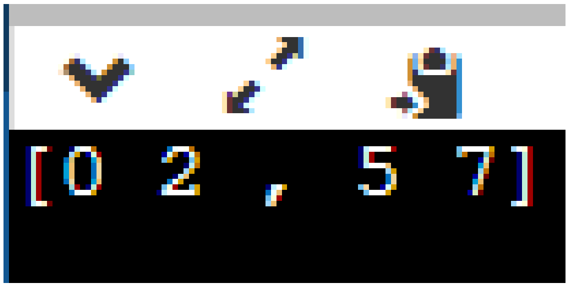
O

D

E

```
#include <iostream>

using namespace std;

class point
{
        public:

        int x;
        int y;

};

class ract
{
    public:

        //ract class data member access
        //point class data member through "::"

        point TL;//point ract :: TL
        point BR;//point ract :: BR

};
int main()
{
    ract r = {{0,2},{5,7}};

    cout<<"["<<r.TL.x<<" "<<r.TL.y<<" , "<<r.BR.x<<" "<<r.BR.y<<"]";

    return 0;
}
```

SCREEN SHOT



```
[0 2 , 5 7]
```

19. **DEFINITION:**

CODE

```
#include <iostream>

using namespace std;
class  Stack {

    public:

        char data[100];
        int top;

        bool empty()
        {
            return (top==-1);
        }

        void push(char x)
        {
            data[top++] = x;

        }
        void pop()
        {
            top--;
        }
};

int main()
{   Stack c;
    cout<<"\n HARDIK DHARAIYA       22FOTCA11034";
    cout<<"\n ----------------------------------\n\n";

    char str[10] = "ABCD";

    int h;

    for(h=0;h<5;h++)
    c.push(str[h]);
    cout<<"\n Reverse String : ";

    while(!c.empty())
    {
        cout<<c.top()<<c.pop();
    }

    return 0;
}
```

**S
C
R
E
E
N**

**S
H
O
T**

20.

DEFINITION:

Operator Overloading

C

O

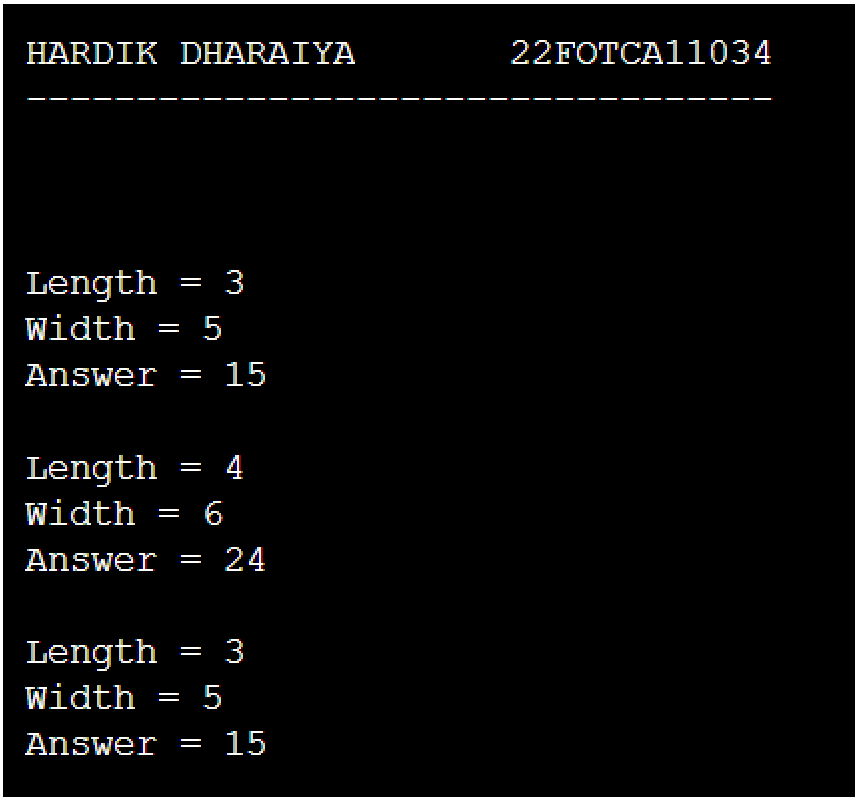
D

E

```

#include <iostream>

using namespace std;
class RACT
{
    public:

    int length;
    int width;


    RACT(int len, int wid)
    {
        length=len;
        width=wid;
    }

    void area()
    {
        cout<<"\n\n Length = "<<length<<"\n Width = "<<width<<"\n Answer =
"<<length*width;
    }

    RACT operator++(int)
    {
        RACT temp=*this;
        length++;
        width++;
        return temp;
    }

    RACT operator--(int)
    {
        RACT temp=*this;
        length--;
        width--;
        return temp;
    }

};
```

```
int main()
{
   RACT r(3,5);

   cout<<"\n HARDIK DHARAIYA      22FOTCA11034";
   cout<<"\n ----------------------------------\n\n";

   r.area();

   r++;

   r.area();


   r--;

   r.area();


   return 0;
}
```

SCREEN SHOT



```
HARDIK DHARAIYA       22FOTCA11034
----------------------------------



Length = 3
Width = 5
Answer = 15

Length = 4
Width = 6
Answer = 24

Length = 3
Width = 5
Answer = 15
```

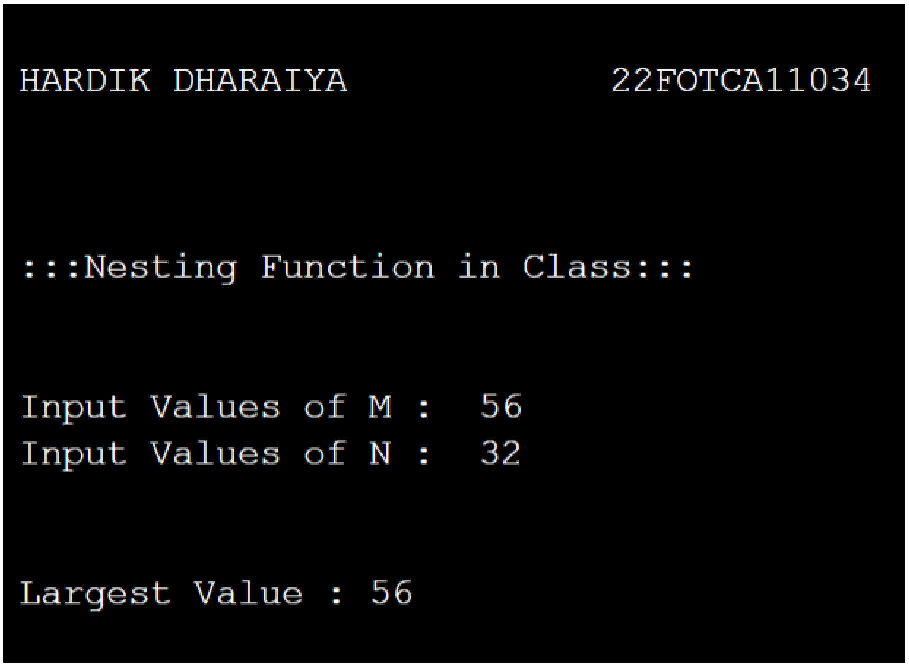
21. Definition :
Nesting Function in Class

C
O
D
E

```
#include <iostream>
using namespace std;
class SET
{
   private:

      int m,n;

   public:

      void input(void);
      void display(void);
      int largest(void);

};

int SET::largest(void)
{
   if(m>n)
   {
      return m;
   }
   else
   {
     return n;
   }
}

void SET::input(void)
{
   cout<<"\n Input Values of M :  ";
   cin>>m;

   cout<<" Input Values of N :  ";
   cin>>n;

}

void SET::display(void)
{
   cout<<"\n\n Largest Value : "<<largest()<<" \n";
}
int main()
{
   cout<<"\n\n HARDIK DHARAIYA          22FOTCA11034 \n\n";

   cout<<"\n\n :::Nesting Function in Class:::\n\n ";

   SET a;

   a.input();

   a.display();

   return 0;
}
```

S
C
R
E
E
N

S
H
O
T



```
HARDIK DHARAIYA            22FOTCA11034



:::Nesting Function in Class:::


Input Values of M :  56
Input Values of N :  32


Largest Value : 56
```

22.	**Definition :** **Nesting Function in Class**
C **O** **D** **E**	
S **C** **R** **E** **E** **N** **S** **H** **O** **T**	

23. Definition :
Operator overloading

C

O

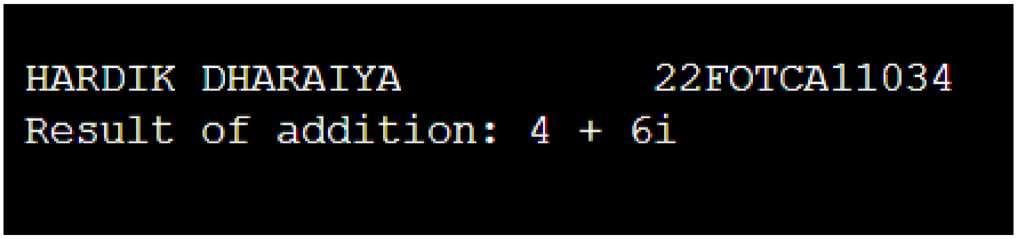
D

E

```
#include <iostream>
using namespace std;

class complex {
private:
    int real;
    int img;

public:
    complex(int h = 0, int r = 0) {
        real = h;
        img = r;
    }

    complex add(complex x) {
        complex temp;
        temp.real = real + x.real;
        temp.img = img + x.img;
        return temp;
    }

    void display() {
        cout << real << " + " << img << "i" << endl;
    }
}; // Missing semicolon here

int main() {
    cout << "\n HARDIK DHARAIYA          22FOTCA11034\n ";

    complex num1(3, 4);
    complex num2(1, 2);

    complex result = num1.add(num2);
    cout << "Result of addition: ";
    result.display();

    return 0;
}
```

S
C
R
E
E
N

S
H
O
T



```
HARDIK DHARAIYA          22FOTCA11034
Result of addition: 4 + 6i
```

24. Definition :

OPerator overloading (name example)

C

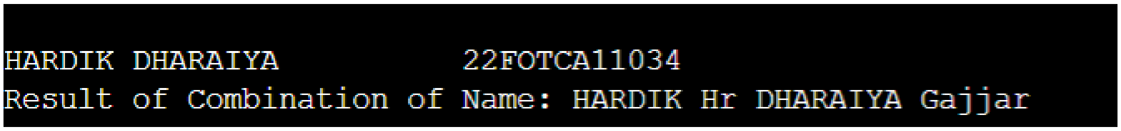
O

D

E

```
#include <iostream>
#include <cstring>
using namespace std;

class name {
private:
    char fname[30];
    char lastname[30];

public:
    name(const char* fn, const char* ln) {
        strcpy(fname, fn);
        strcpy(lastname, ln);
    }

    // Combine the first name and second name
    name add(const name& x) {
        name temp("", ""); // Initialize with empty strings
        strcat(temp.fname, fname); // Concatenate strings
        strcat(temp.fname, " ");   // Add space between first and last names
        strcat(temp.fname, x.fname);
        strcat(temp.lastname, lastname);
        strcat(temp.lastname, " ");
        strcat(temp.lastname, x.lastname);
        return temp;
    }

    void display() {
        cout << fname << " " << lastname << endl;
    }
};

int main() {
    cout << "\n HARDIK DHARAIYA          22FOTCA11034\n ";

    name name1("HARDIK", "DHARAIYA");
    name name2("Hr", "Gajjar");

    name result = name1.add(name2);
    cout << "Result of Combination of Name: ";
    result.display();
    return 0; }
```

SCREEN SHOT



```
HARDIK DHARAIYA          22FOTCA11034
Result of Combination of Name: HARDIK Hr DHARAIYA Gajjar
```

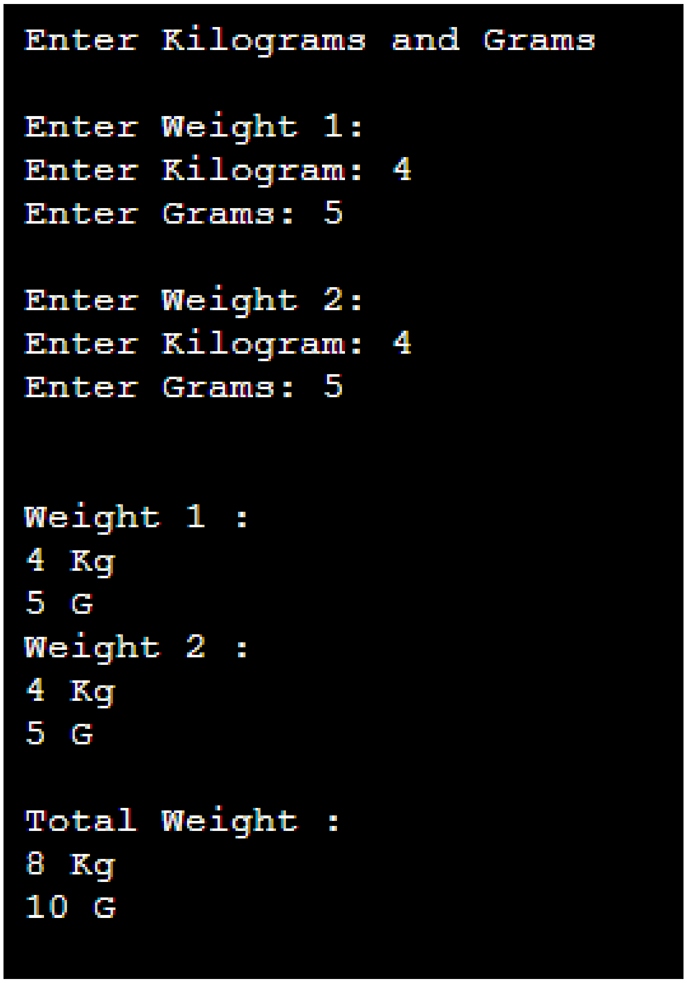
25.	**Definition :** **OPerator overloading ()**
CODE	
SCREEN SHOT	

26. Definition :
Object as Argument in function

C
O
D
E

```
#include <iostream>
using namespace std;
class OBJECT
{
        int kg;
        int g;
    public:
        void getdata();
        void display();

        OBJECT sum(OBJECT o1, OBJECT o2)
        {
            OBJECT o3;

            o3.g = o1.g + o2.g;
            o3.kg = o1.kg + o2.kg;
            return o3;

        }
};

void OBJECT::getdata()
{

    cout<<"\n Enter Kilogram: ";
    cin>>kg;

    cout<<" Enter Grams: ";
    cin>>g;

}
void OBJECT::display()
{
    cout<<"\n "<<kg<<" Kg \n "<<g<<" G" ;
}
int main()
{

    OBJECT k1, k2, k3;

    cout<<"\n Enter Kilograms and Grams\n ";
    cout<<"\n Enter Weight 1: ";
    k1.getdata();
    cout<<"\n Enter Weight 2: ";
    k2.getdata();

    k3 = k3.sum(k1, k2);

    cout<<"\n\n Weight 1 : ";
    k1.display();

    cout<<"\n Weight 2 : ";
    k2.display();

    cout<<"\n\n Total Weight : ";
    k3.display();

    return 0;
}
```

SCREEN SHOT



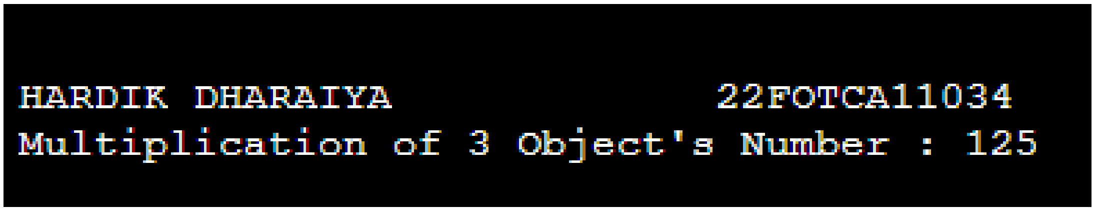
```
Enter Kilograms and Grams

Enter Weight 1:
Enter Kilogram: 4
Enter Grams: 5

Enter Weight 2:
Enter Kilogram: 4
Enter Grams: 5


Weight 1 :
4 Kg
5 G
Weight 2 :
4 Kg
5 G

Total Weight :
8 Kg
10 G
```

27. Definition : Friend function

CODE

```
#include <iostream>
using namespace std;
class H;
class R;
class J;

class H
{
        int num;

    public:
        H(int x)
        {
            num=x;
        }

        friend int multi(H, R, J);
};

class R
{
        int num;

    public:
        R(int x)
        {
            num=x;
        }

        friend int multi(H, R, J);
};

class J
{
        int num;
    public:
        J(int x)
        {
            num=x;
        }

        friend int multi(H, R, J);
};

int multi(H ob1, R ob2, J ob3)
{
    return (ob1.num * ob2.num * ob3.num);
}

int main()
{
    int ans;

    H o1(5);
    R o2(5);
    J o3(5);

    cout<<"\n\n HARDIK DHARAIYA            22FOTCA11034";
    ans = multi(o1, o2, o3);

    cout<<"\n Multiplication of 3 Object's Number : "<<ans;

}
```

S
C
R
E
E
N



```
HARDIK DHARAIYA                         22FOTCA11034
Multiplication of 3 Object's Number : 125
```

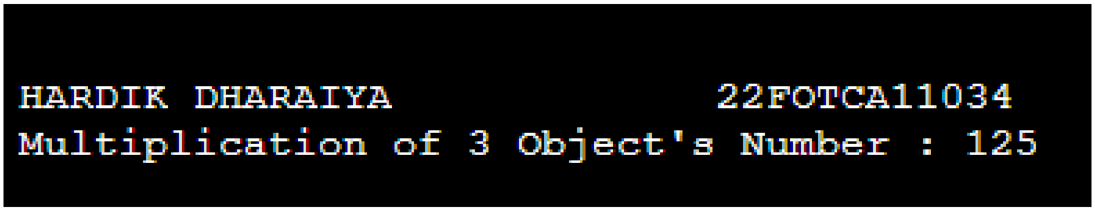
27. Definition :
Friend function

CODE

```
#include <iostream>
using namespace std;
class H;
class R;
class J;

class H
{
        int num;

    public:
        H(int x)
        {
            num=x;
        }

        friend int multi(H, R, J);
};

class R
{
        int num;

    public:
        R(int x)
        {
            num=x;
        }

        friend int multi(H, R, J);
};

class J
{
        int num;
    public:
        J(int x)
        {
            num=x;
        }

        friend int multi(H, R, J);
};
```

```
// define the friend function
int multi(H ob1, R ob2, J ob3)
{
   return (ob1.num * ob2.num * ob3.num);
}

int main()
{
   int ans;

   H o1(5);
   R o2(5);
   J o3(5);

   cout<<"\n\n HARDIK DHARAIYA            22FOTCA11034";
   ans = multi(o1, o2, o3);

   cout<<"\n Multiplication of 3 Object's Number : "<<ans;

}
```

SCREEN



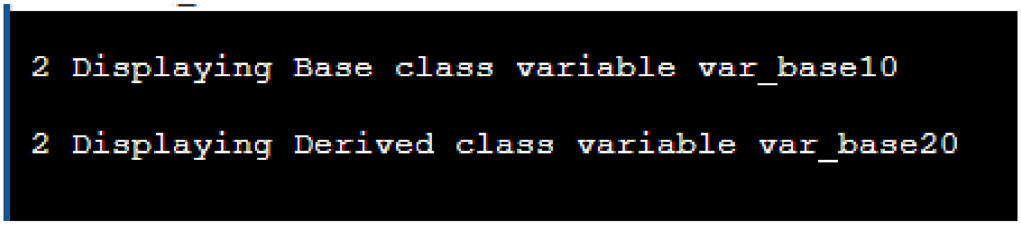
```

HARDIK DHARAIYA                 22FOTCA11034
Multiplication of 3 Object's Number : 125
```

28. Definition :
Virtual Function

CODE

```
#include<iostream>
using namespace std;

class BaseClass
{
   public:
      int var_base=10;
      virtual void display()
      {
         cout<<"1 displaying Base class variable var_base"<<var_base<<endl;
      }
};

class DerivedClass : public BaseClass
{
   public:
      int var_derived=20;
      void display()
   {
      cout<<"\n 2 Displaying Base class variable var_base"<<var_base<<endl;
      cout<<"\n 2 Displaying Derived class variable var_base"<<var_derived<<endl;
   }
};

int main()
{
      BaseClass * base_class_pointer;
      BaseClass obj_base;
      DerivedClass obj_derived;

      base_class_pointer = &obj_derived;
      base_class_pointer->display();
return 0;
}
```

SCREEN



```
 2 Displaying Base class variable var_base10

 2 Displaying Derived class variable var_base20
```

29. Definition : Pure Virtual Function

C

O

D

E

```
#include<iostream>
using namespace std;
class shape
{
    protected:
        float dimenstion;

    public:
        void getDimenstion()
        {
            cout<<"\n Enter the number: ";
            cin>>dimenstion;
        }

        virtual float calculateArea() = 0;
};
class square : public shape
{
    public:
        float calculateArea()
        {
            return dimenstion*dimenstion;
        }

};

class circle : public shape
{
    public:
        float calculateArea()
        {
            return 3.14 * dimenstion * dimenstion;
        }
}
int main()
{
    square s;
    circle c;
```

	return 0;
	}

S C R E E N	