Name: Owais Ali Khan, Roll no: 21K-3298 *Section: F*

AP LAB ASSIGNMENT:

Question 1: Write a program which takes the upper bound value from user and display the perfect squares between 1 and the upper bound.

Definition: Perfect squares are numbers whose square roots are whole numbers e.g:[1,4,9,16,25].

Sample Input:

Enter The Upper Bound: 133

Sample Output:

The Perfect Squares Are:

1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121

In [3]:
```
def perfect_square(n):
    x = 0
    #print("[", end='')
    while x**2 < n:
        #print(a, end=',')
        if (x+1)**2 > n:
            print(x**2, end='')
        else:
            print(x**2, end=', ')
        x += 1
n = int(input("Enter The Upper Bound: "))
print("The Perfect Squares Are:")
perfect_square(n)
```

```
Enter The Upper Bound: 133
The Perfect Squares Are:
0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121
```

Name: Owais Ali Khan, Roll no: 21K-3298 *Section: F*

Question 2: Write a python program that takes input from the user and pass this input into a function called fibonacci(n) this function prints the n terms of fibonacci series and return your roll number.

print your roll number in main function.

print your name and section first then print the fibonacci series.

Sample Input:

Enter Number Of values: 7

Sample Output:

My Name Is Taaha My section is B

Fibo Series:

[0 1 1 2 3 5 8]

My Roll Number Is: returned Value(e.g 210000)

In [4]:
```
name = "Owais"
section = "F"
def fibo(n):
    #i = 1000
    #a,b = 0,1
    a = 0
    b = 1
    print("[", end='')
    for x in range(n):
        #print(a, end=',')
        #if x > n-2:
        if x == n-1:
            print(a, end='')
        else:
            print(a, end=' ')
        a,b = b,a+b
    print("]")
    return "21K-3298"
n = int(input("Enter Number Of values: "))
#n=7
print(f'My name is {name} and my section is {section}')
print("Fibo Series:")
#fibo(n)
print("My roll number is:",fibo(n))
```

```
Enter Number Of values: 7
My name is Owais and my section is F
Fibo Series:
[0 1 1 2 3 5 8]
My roll number is: 21K-3298
```

Name: Owais Ali Khan, Roll no: 21K-3298 *Section: F*

Question 3: A ball is dropped from a tower of height h. It has initial velocity zero and accelerates downwards under gravity. The challenge is to write a program that asks the user to enter the height in meters of the tower and a time interval t in seconds, then prints on the screen the height of the ball above the ground at time t after it is dropped, ignoring air resistance. The steps involved are the following. First, we will use input statements to get the values of h and t from the user. Second, we will calculate how far the ball falls in the given time, using the standard kinematic formula:

$$s=1/2gt^2$$

where g = 9.81 ms^{-2} is the acceleration due to gravity. Third, we print the height above the ground at time t, which is equal to the total height of the tower minus this value. Write a python function to calculate the height of a ball after the t seconds.
Hint: Height(h,t) ► return h-s
Instruction: Take input height as your roll number E.g. if your roll number is 19K1328 then take 1328 as height.
Sample Input:
Enter the height of the tower: 1328
Enter the time interval: 9
Sample Output
The height of the ball is 930.6949 meters

In [5]:
```
height = int(input("Enter height of the tower: "))
time = int(input("Enter the time interval: "))

dist = 1/2 * 9.81 * time**2
ball = height - dist
print(f'The height of the ball is {ball:.4f} meters')
```

```
Enter height of the tower: 1328
Enter the time interval: 9
The height of the ball is 930.6950 meters
```

Name: Owais Ali Khan, Roll no: 21K-3298 *Section: F*

Question 4: Plot two vectors and their resultant vector using matplotlib library functions in Python.

In [6]:
```
import numpy as np
import matplotlib.pyplot as plt

#origin = np.array([[0, 0, 0],[0, 0, 0]]) # origin point
v1 = np.array([3,5])
v2 = np.array([6,-4])
r = v1 + v2
fig, ax = plt.subplots()
plt.quiver(0,0, v1[0],v1[1], color='r',angles='xy', scale_units='xy', scale=1, label="Vector 1")
plt.quiver(0,0, v2[0],v2[1], color='b',angles='xy', scale_units='xy', scale=1, label="Vector 2")
plt.quiver(0,0, r[0],r[1], color='g',angles='xy', scale_units='xy', scale=1, label="Resultant")
ax.axis([-10, 10, -10, 10])
ax.grid(b=True, which='major')
ax.set_title('Resolution of two vectors')
plt.xlabel("X axis")
plt.ylabel("Y axis")
ax.legend()
plt.show()
```