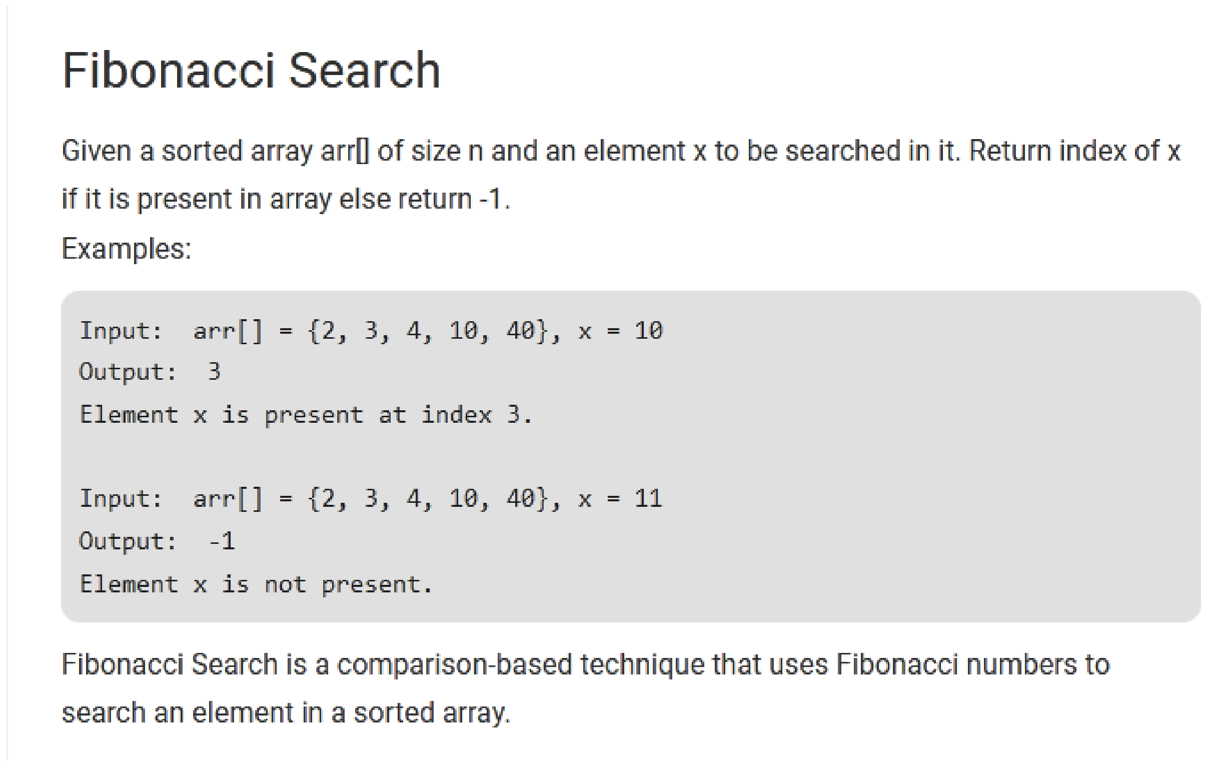
Assignment No. 4B

Write a python program to store roll numbers of student array who attended training program in sorted order. Write function for searching whether particular student attended training program or not, using Binary search and Fibonacci search

PROBLEM STATEMENT:

B) WRITE A PYTHON PROGRAM TO STORE ROLL NUMBERS OF STUDENT ARRAY WHO ATTENDED TRAINING PROGRAM IN SORTED ORDER. WRITE FUNCTION FOR SEARCHING WHETHER PARTICULAR STUDENT ATTENDED TRAINING PROGRAM OR NOT, USING BINARY SEARCH AND FIBONACCI SEARCH



Fibonacci Search

Given a sorted array arr[] of size n and an element x to be searched in it. Return index of x if it is present in array else return -1.

Examples:

```
Input:  arr[] = {2, 3, 4, 10, 40}, x = 10
Output:  3
Element x is present at index 3.

Input:  arr[] = {2, 3, 4, 10, 40}, x = 11
Output:  -1
Element x is not present.
```

Fibonacci Search is a comparison-based technique that uses Fibonacci numbers to search an element in a sorted array.

Similarities with Binary Search:

1. Works for sorted arrays
2. A Divide and Conquer Algorithm.
3. Has Log n time complexity.

Differences with Binary Search:

1. Fibonacci Search divides given array in unequal parts
2. Binary Search uses division operator to divide range. Fibonacci Search doesn't use /, but uses + and -. The division operator may be costly on some CPUs.
3. Fibonacci Search examines relatively closer elements in subsequent steps. So when input array is big that cannot fit in CPU cache or even in RAM, Fibonacci Search can be useful.

Background:

Fibonacci Numbers are recursively defined as F(n) = F(n-1) + F(n-2), F(0) = 0, F(1) = 1. First few Fibinacci Numbers are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

Observations:

Below observation is used for range elimination, and hence for the O(log(n)) complexity.

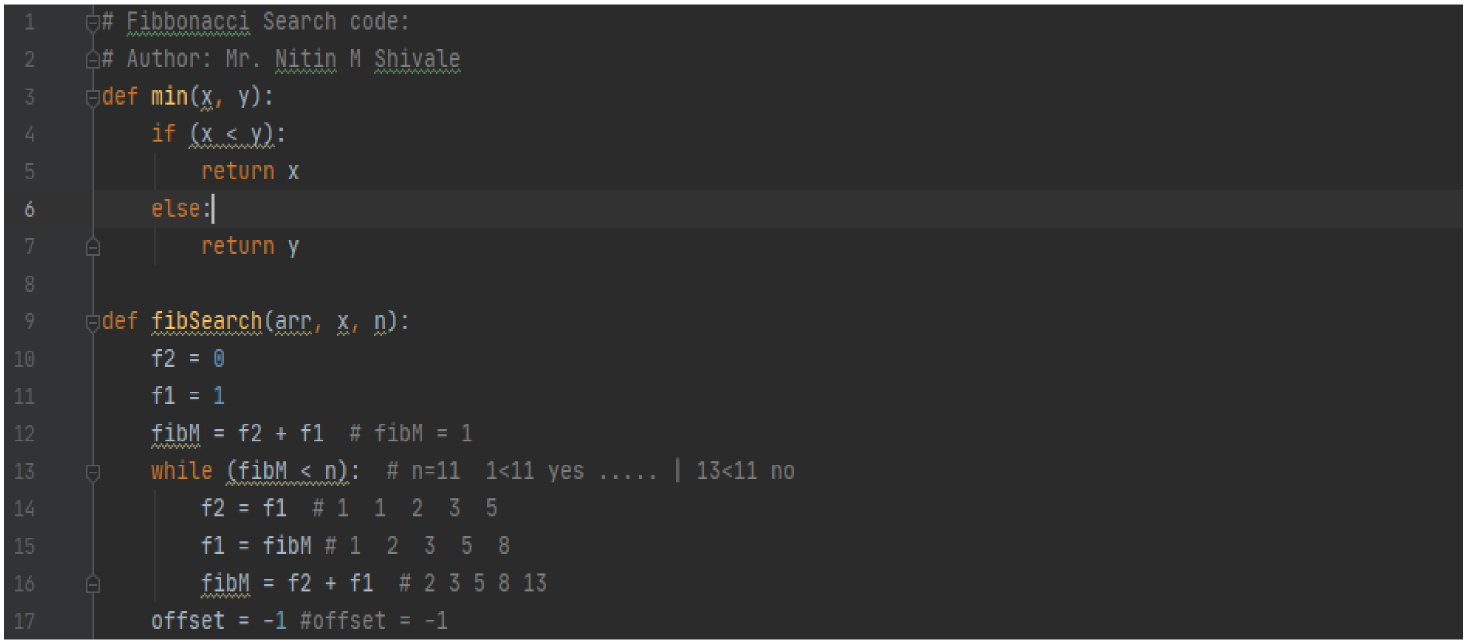
```
F(n - 2) &approx; (1/3)*F(n) and
F(n - 1) &approx; (2/3)*F(n).
```

Algorithm:
Let the searched element be x.

The idea is to first find the smallest Fibonacci number that is greater than or equal to the length of given array. Let the found Fibonacci number be fib (m'th Fibonacci number). We use (m-2)'th Fibonacci number as the index (If it is a valid index). Let (m-2)'th Fibonacci Number be i, we compare arr[i] with x, if x is same, we return i. Else if x is greater, we recur for subarray after i, else we recur for subarray before i.

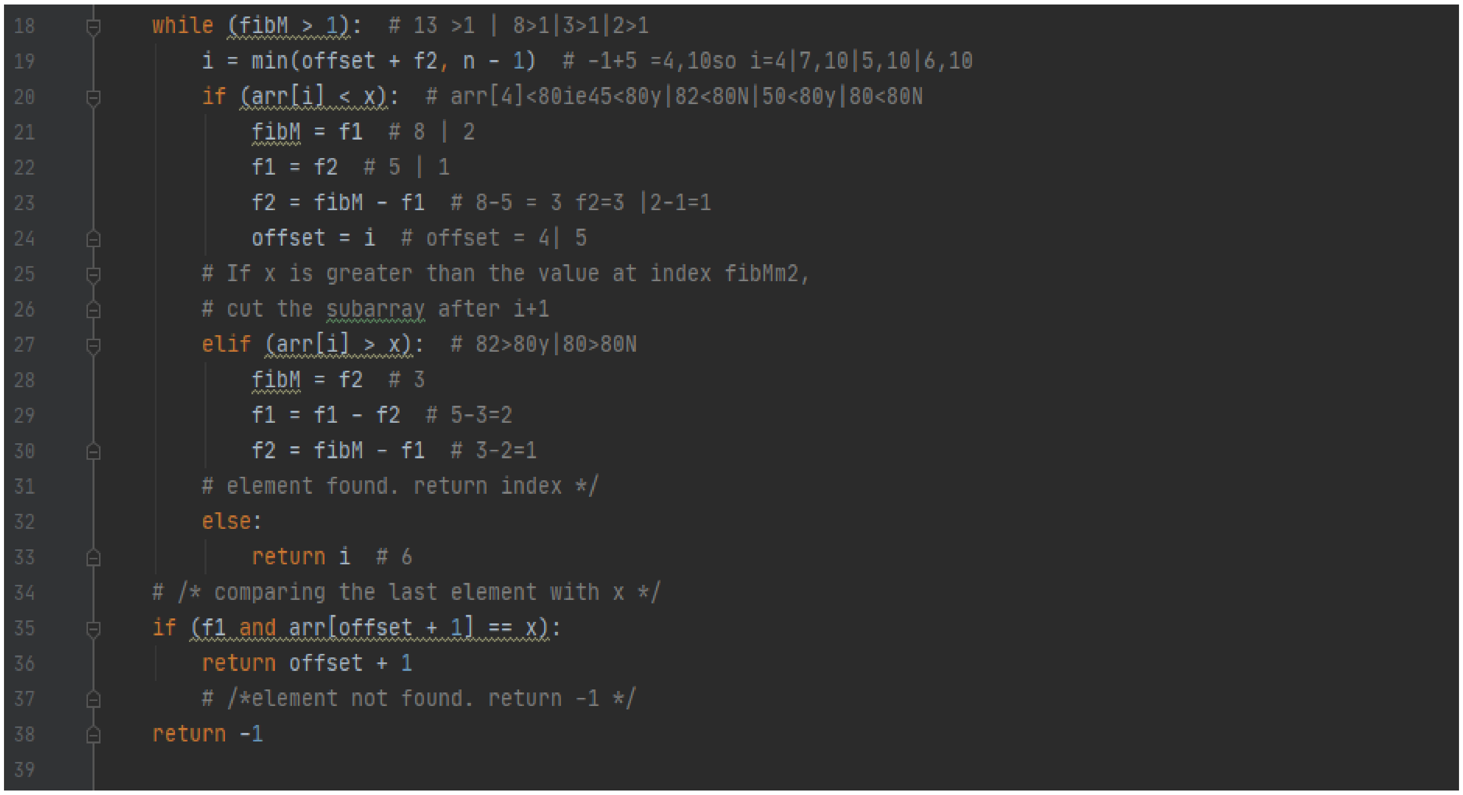
Below is the complete algorithm
Let arr[0..n-1] be the input array and element to be searched be x.

1. Find the smallest Fibonacci Number greater than or equal to n. Let this number be fibM [m'th Fibonacci Number]. Let the two Fibonacci numbers preceding it be fibMm1 [(m-1)'th Fibonacci Number] and fibMm2 [(m-2)'th Fibonacci Number].
2. While the array has elements to be inspected:
    1. Compare x with the last element of the range covered by fibMm2
    2. **If** x matches, return index
    3. **Else If** x is less than the element, move the three Fibonacci variables two Fibonacci down, indicating elimination of approximately rear two-third of the re-maining array.
    4. **Else** x is greater than the element, move the three Fibonacci variables one Fibonacci down. Reset offset to index. Together these indicate elimination of approximately front one-third of the remaining array.
3. Since there might be a single element remaining for comparison, check if fibMm1 is 1. If Yes, compare x with that remaining element. If match, return index.

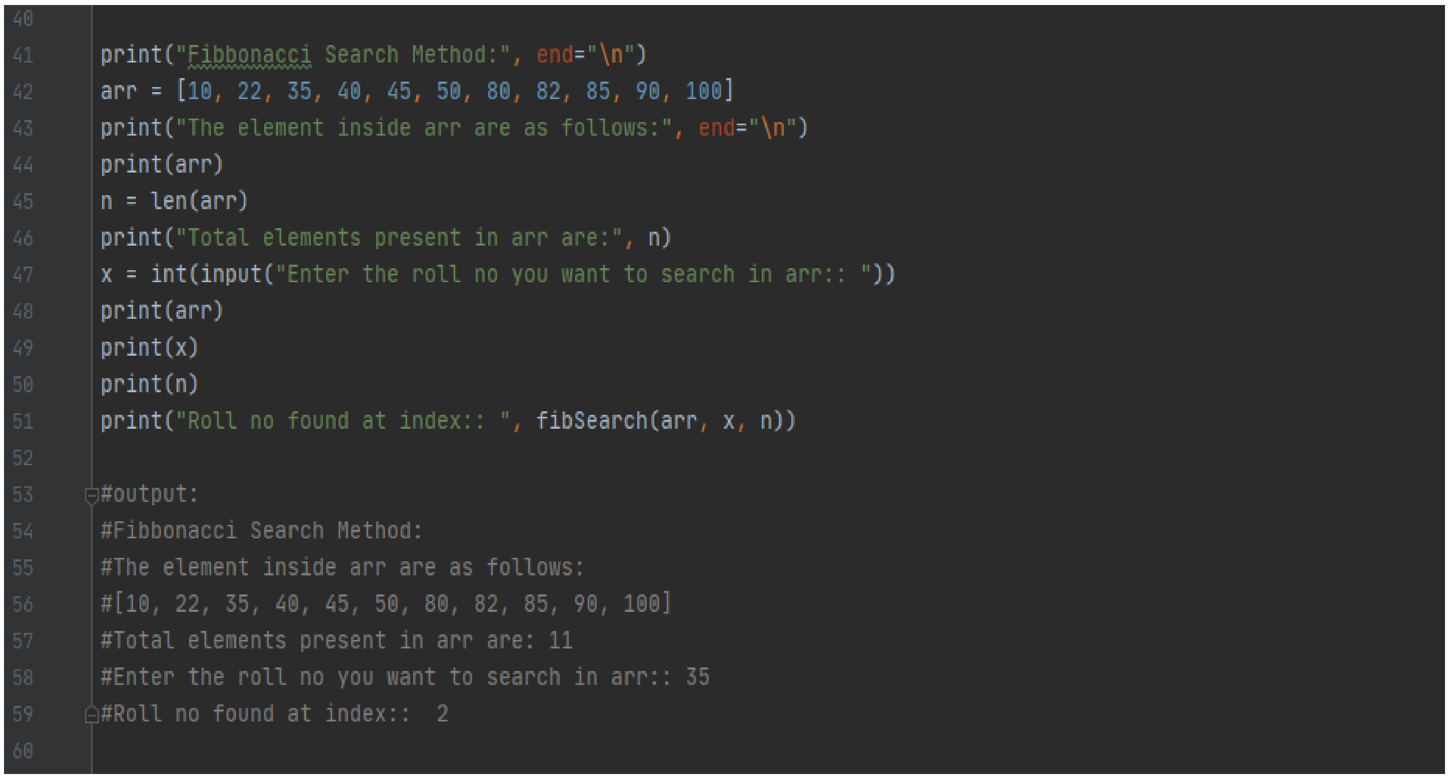


```python
# Fibbonacci Search code:
# Author: Mr. Nitin M Shivale
def min(x, y):
    if (x < y):
        return x
    else:
        return y

def fibSearch(arr, x, n):
    f2 = 0
    f1 = 1
    fibM = f2 + f1  # fibM = 1
    while (fibM < n):  # n=11  1<11 yes ..... | 13<11 no
        f2 = f1  # 1  1  2  3  5
        f1 = fibM # 1  2  3  5  8
        fibM = f2 + f1  # 2 3 5 8 13
    offset = -1 #offset = -1
```



```python
    while (fibM > 1):  # 13 >1 | 8>1|3>1|2>1
        i = min(offset + f2, n - 1)  # -1+5 =4,10so i=4|7,10|5,10|6,10
        if (arr[i] < x):  # arr[4]<80ie45<80y|82<80N|50<80y|80<80N
            fibM = f1  # 8 | 2
            f1 = f2  # 5 | 1
            f2 = fibM - f1  # 8-5 = 3 f2=3 |2-1=1
            offset = i  # offset = 4| 5
        # If x is greater than the value at index fibMm2,
        # cut the subarray after i+1
        elif (arr[i] > x):  # 82>80y|80>80N
            fibM = f2  # 3
            f1 = f1 - f2  # 5-3=2
            f2 = fibM - f1  # 3-2=1
        # element found. return index */
        else:
            return i  # 6
    # /* comparing the last element with x */
    if (f1 and arr[offset + 1] == x):
        return offset + 1
        # /*element not found. return -1 */
    return -1
```



```python

print("Fibbonacci Search Method:", end="\n")
arr = [10, 22, 35, 40, 45, 50, 80, 82, 85, 90, 100]
print("The element inside arr are as follows:", end="\n")
print(arr)
n = len(arr)
print("Total elements present in arr are:", n)
x = int(input("Enter the roll no you want to search in arr:: "))
print(arr)
print(x)
print(n)
print("Roll no found at index:: ", fibSearch(arr, x, n))

#output:
#Fibbonacci Search Method:
#The element inside arr are as follows:
#[10, 22, 35, 40, 45, 50, 80, 82, 85, 90, 100]
#Total elements present in arr are: 11
#Enter the roll no you want to search in arr:: 35
#Roll no found at index::  2

```