**KGiSL INSTITUTE OF TECHNOLOGY: COIMBATORE**

(Approved by AICTE & Affiliated to Anna University, Chennai)

**GE8161**

**Problem Solving and**

**Python Programming Laboratory**

LAB PREPARATION GUIDE

Department of Computer Science and Engineering

**KGiSL INSTITUTE OF TECHNOLOGY**

Saravanampatti, Coimbatore-35

**GE8161 PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY**

**LIST OF PROGRAMS**

1. Compute the GCD of two numbers.
2. Find the square root of a number (Newton‘s method)
3. Exponentiation (power of a number)
4. Find the maximum of a list of numbers
5. Linear search and Binary search
6. Selection sort, Insertion sort
7. Merge sort
8. First n prime numbers
9. Multiply matrices
10. Programs that take command line arguments (word count)
11. Find the most frequent words in a text read from a file
12. Simulate elliptical orbits in Pygame
13. Simulate bouncing ball using Pygame

|  |  |
| --- | --- |
| **Ex No: 1** | COMPUTE THE GCD OF TWO NUMBERS |

**AIM**

Write a Python program to compute the greatest common divisor (***gcd***) of two positive integers.

**ALGORITHM**

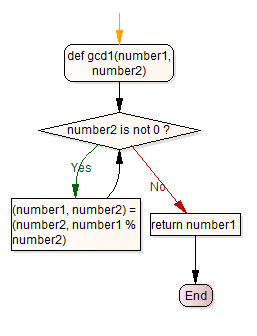
Function: **GCD**

**Parameters**: number1, number2

Step 1: while number2 is not 0,   
 Step 1a: number1, number2 = number2, number1 % number2

Step 2: return number1 as GCD

**FLOWCHART**



**PROGRAM**

def gcd(number1, number2):

while True:

if number2 != 0:

(number1, number2) = (number2, number1 % number2)

else:

break

return number1

**TESTING**

a = 8

b = 6

print ('The gcd is ', gcd(a, b))

**OUTPUT**

12

24

The gcd is 12

**RESULT**

Thus the python program to compute the greatest common divisor (***gcd***) of two positive integers has been written, executed and verified successfully.

|  |  |
| --- | --- |
| **Ex No: 2** | THE SQUARE ROOT (Newton’s Method) |

**AIM**

Write a Python program to Implement a python program that determines the square root of a number using the Newton's method.

**ALGORITHM**

Step 1: Initialize  
 epsilon = 0.01

k = number

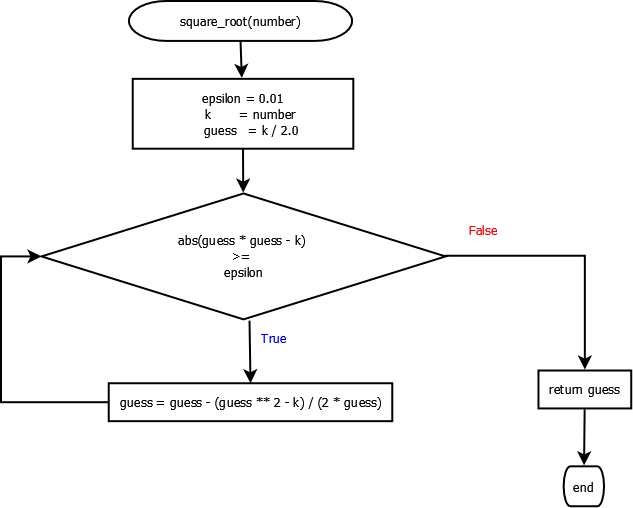
guess = k / 2.0

Step 2: while (abs(guess \* guess- k) >= epsilon) :

guess = guess - (guess \*\*2 - k) / (2 \* guess)

Step 3: return guess

**FLOWCHART**

****

**PROGRAM**

def square\_root(number):

epsilon = 0.01

k = number

guess = k / 2.0

while abs(guess \* guess - k) >= epsilon:

guess = guess - (guess \*\* 2 - k) / (2 \* guess)

return guess

**TESTING**

a = int(input())

print (square\_root(a))

**OUTPUT**

2

1.4166666666666667

**RESULT**

Thus the python program to compute the square root of a number using Newton’s method has been written, executed and verified successfully.

|  |  |
| --- | --- |
| **Ex No: 3** | EXPONENTIATION (POWER OF A NUMBER) |

**AIM**

Write a Python program to compute the power of a given number.

**ALGORITHM**

Function: **EXPONENT**

**Parameters**: base, exp

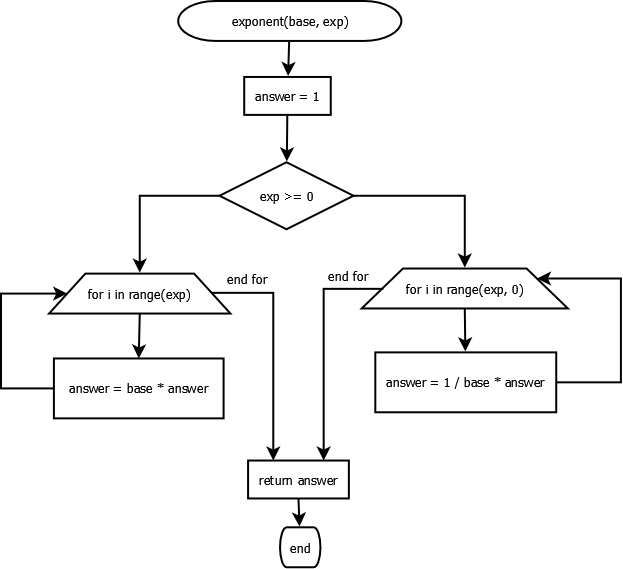
Step 1: Initialize  
 answer = 1.0

Step 2: if the input parameters are not valid,   
 return “invalid input”  
Step 3: if ‘exp’ is positive number (>=0),  
 multiply base ‘exp’ times and store it in ‘answer’

else,  
 multiply 1/base ‘exp’ times and store it in ‘answer’

Step 4: return the ‘answer’

**FLOWCHART**

****

**PROGRAM**

def exponent(base, expo):

answer = 1.0

if exp >= 0:

for i in range(exp):

answer = base \* answer

else:

for i in range(expo, 0):

answer = 1/base \* answer

return answer

**TESTING**

print (exponent(4, 3))

**OUTPUT**

64.0

**RESULT**

Thus the python program to compute the number to the power of exp (***exponentiation***) has been written, executed and verified successfully.

|  |  |
| --- | --- |
| **Ex No: 4** | FIND THE MAXIMUM OF A LIST OF NUMBERS |

**AIM**

Implement a python program that finds the maximum in a list of numbers.

**ALGORITHM**

Function: **get\_max**

**Parameters**: L

Step 1: maxVal = None

Step 2: for number in L

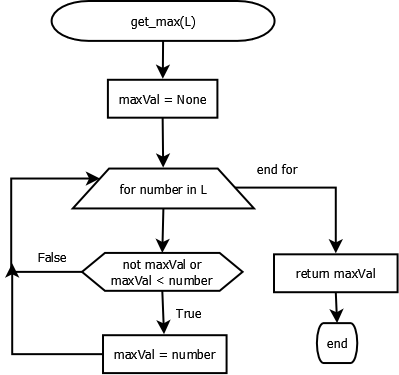
Step 2.1 : If maxval is not set, update with number

Step 2.1 : If maxVal is not set or maxVal is less than number,

maxVal = number

Step 3: return maxVal

**FLOWCHART**



**PROGRAM**

def get\_max(L):

maxval = None

for num in numbers:

if not maxval or maxval < num:

maxval = num

return maxval

**TESTING**

L = [1, 4, 6, 7]

print (get\_max(L))

**OUTPUT**

7

**RESULT**

Thus the python program to find the maximum number from the given list has been written, executed and verified successfully.

|  |  |
| --- | --- |
| **Ex No: 5** | LINEAR SEARCH AND BINARY SEARCH |

**AIM**

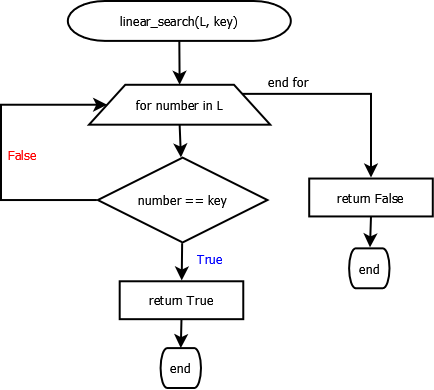
To write a python program for performing linear search and binary search for the given set of inputs

LINEAR SEARCH

**ALGORITHM**Function: linear\_search  
parameters: L, key  
 **Step 1:** for number in L:  
 **Step 1.1:** if number == key:  
 return True

**Step 2:** return False

**FLOWCHART**



**PROGRAM**

def linear\_search(L, key):

for number in L:

if number == key:

return True

return False

**TESTING**

L = [1, 4, 6, 7]

print (linear\_search(L, 20))

print (linear\_search(L, 6))

**OUTPUT**

False  
True

BINARY SEARCH

**ALGORITHM**

**Step1:** Choose the middle element in the list

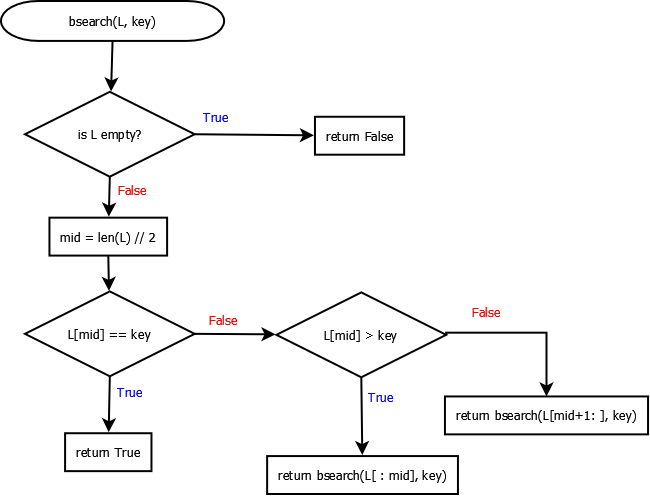
**Step2:**  If it matches the middle element, its position in the list is returned.

**Step 3:** If the target value is less than or greater than the middle element, the search

continues in the lower or upper half of the array, respectively, eliminating the

other half from consideration

**FLOWCHART**



**PROGRAM**

def bsearch(L, key):

if not L:

return False

mid = len(L) // 2

if L[mid] == key:

return True

elif L[mid] > key:

return bsearch(L[:mid], key)

else:

return bsearch(L[mid+1:], key)

**TESTING**

L = [1, 4, 6, 7]

print (bsearch(L, 20))

print (bsearch(L, 6))

**OUTPUT**

False  
True

**RESULT**

Thus the python program for performing the linear and binary search was executed and verified successfully.

|  |  |
| --- | --- |
| **Ex No: 6** | SELECTION SORT AND INSERTION SORT |

**AIM**

To write a python program for sorting the given set of inputs using selection sort and insertion sort

SELECTION SORT

**ALGORITHM**

**Step 1:** Get started from the first element (i = 0): **Step1.1:** Get the unsorted sub list (U)

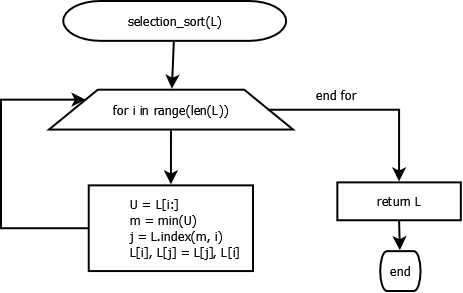
**Step1.2:**  Find the smallest in the sublist (m)

**Step 1.3:** Find the index of the smallest (j)

**Step 1.4:** Swap the smallest with the element at index (i)

**Step 2:** Increment the index (i) and repeat step 1 until list is sorted

**FLOWCHART**



**PROGRAM**

def selectsort(L):

for i in range(len(L)):

U = L[i:]

m = min(U)

j = L.index(m, i)

L[i], L[j] = L[j], L[i]

return L

**TESTING**

L = [12, 4, 7, 56, 2]

selectsort(L)

print(L)

**OUTPUT**

[2, 4, 7, 12, 56]

INSERTION SORT

**ALGORITHM**

**Step1:** Get the List(L) to be sorted

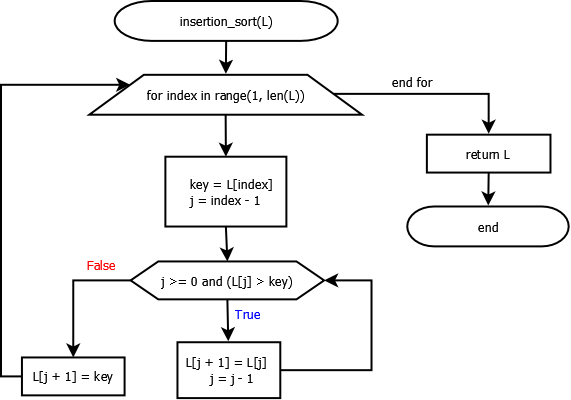
**Step2:** Iterate index from 1 to len(L)

**Step2.1:**  j = index -1  
 key = L[index]   
**Step 2.2:** Until j > 0 and L[j] > key:  
 Step 2.2.1: L[j+1] = L[j]  
 Step 2.2.2: Decrement j by 1

**Step 2.3:** Insert the key at L[j+1]

**Step 3:** Return the sorted list (L)

**FLOWCHART**



**PROGRAM**

def insertsort(L):

for index in range(1, len(L)):

key = L[index]

j = index - 1

while j >= 0 and (L[j] > key):

L[j + 1] = L[j]

j = j - 1

L[j + 1] = key

return L

**# Test**

print(insertsort([5, 4, 0, 29, 2]))

**OUTPUT**

[ 0, 2, 4, 5, 29 ]

**RESULT**

Thus the python program for performing the selection and insertion sort was executed and verified successfully.

|  |  |
| --- | --- |
| **Ex No: 7** | MERGE SORT |

**AIM**

To write a python program to sort the given list using merge sort algorithm.

**ALGORITHM**

**Step1:** If the list (L) is empty or the list with single element,   
 **return L**

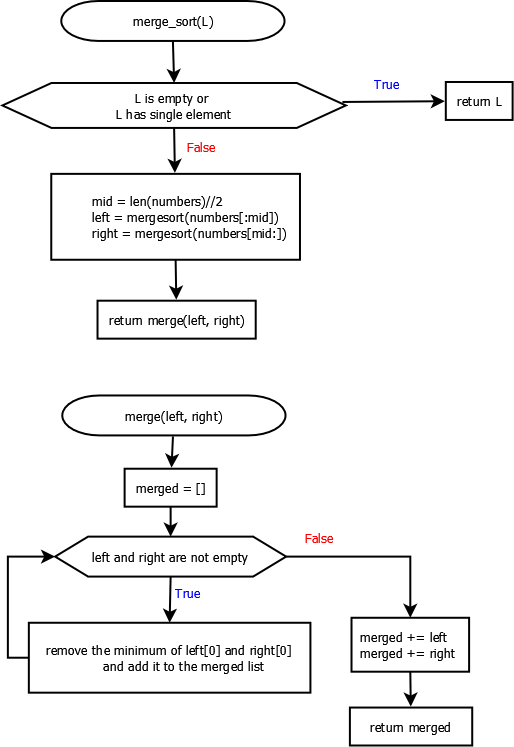
**Step2:**  Recursively split the list into left and right halves, till it contains single element

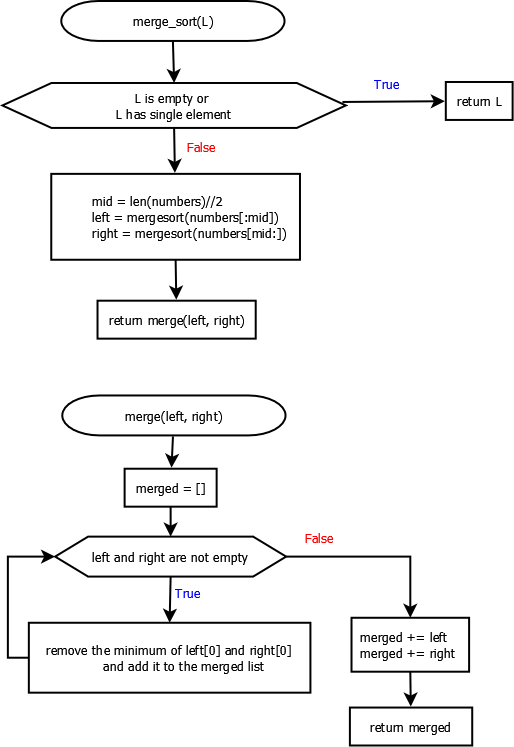
**Step 3:** Merge( ) the ‘left’ and ‘right’ portions in the sorted order to the way up.

**merge( left, right):**

**Step 1:** Get thesorted lists`left` and `right` to be merged.  
**Step 2:** Initialize the empty `merged` list  
**Step 3:** Loop till `Left` or `right` becomes empty:  
 Step 3.1: Remove the minimum of the left[0] and right[0]  
 and add it to the merged list  
**Step 4:** Add the remaining elements of left/ right to the `merged` list  
**Step 5:** Return the `merged` list

**FLOWCHART**





**PROGRAM**

def mergesort(numbers):

if not numbers or len(numbers) == 1:

return numbers

else:

mid = len(numbers)//2

left = mergesort(numbers[:mid])

right = mergesort(numbers[mid:])

return merge(left, right)

def merge(left, right):

merged = []

while left and right:

if left[0] < right[0]:

merged += [left.pop(0)]

else:

merged += [right.pop(0)]

merged += left

merged += right

return merged

**TESTING**

L = [12, 4, 7, 56, 2]

selectsort(L)

print(L)

**OUTPUT**

[2, 4, 7, 12, 56]

**RESULT**

Thus the python program for performing the merge sort was executed and verified successfully.

|  |  |
| --- | --- |
| **Ex No: 8** | FIRST N PRIME NUMBERS |

**AIM**

To write a python program to find the first n prime numbers

**ALGORITHM**

**generate\_primes(min, max)**

**Step 1:** Create the empty list ‘primes’  
**Step 2:** For number in range(min, max)

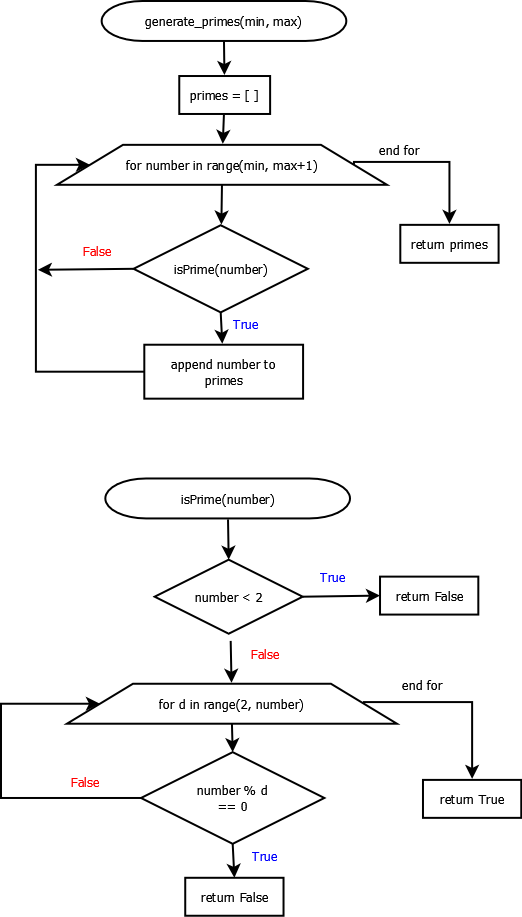
**Step 2.1:** Check whether number is prime using isPrime().   
 If number is prime, append number to primes list

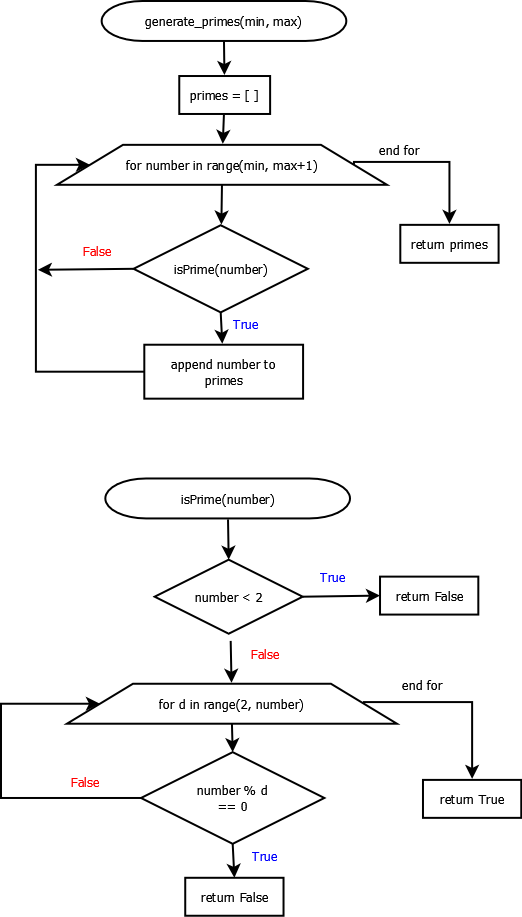
**Step 3**: return primes

**isPrime(number)**

**Step 1:** if number < 2, return False  
**Step 2:** for d in range(2, number):  
 **Step 2.1:** if number is divisible by d,  
 return False  
**Step 3:** return True

**FLOWCHART**





**PROGRAM**

def generate\_primes(min, max):

primes = [ ]

for num in range(min, max + 1):

if isPrime(num):

primes.append(num)

return primes

def isPrime(num):

if num < 2:

return False

for i in range(2, num):

if num % i == 0:

return False

return True

**OUTPUT**

Enter number:15

Output:

2

3

5

7

11

13

**RESULT**

Thus the python program for finding the first n prime numbers was executed and verified successfully.