Algorithms

1)bubble sort algorithm;

Bubble Sort:
Bubble sort is a comparison algorithm that first compares and then sorts adjacent elements if they are not in the specified order.
Bubble Sort Algorithm:
Step 1: Beginning from the first element i.e the element at index 0, progressively compare adjacent elements of an array
Step 2: If the current element and the next element are not in the specified order, swap the elements
Step 3: If the current element and the next element are in the specified order, move on to the next element

Bubble Sort Program:

```
1  def bs(a):                # a = name of list
2      b=len(a)-1
3      for x in range(b):
4          for y in range(b-x):
5              if a[y]>a[y+1]:
6                  a[y],a[y+1]=a[y+1],a[y]
7      return a
8  a=[3,6,1,8]
9  bs(a)
```

OUTPUT: [1, 3, 6, 8]

2)Quick sort: more efficient sorting alorutham then bubble sort as it uses divide and conquer , partition based on pivot element that could be t start, end, middle or random.
When pivot is at the beginning

```
def sort(array=[12,4,5,6,7,3,1,15]):
    less = []
    equal = []
    greater = []

    if len(array) > 1:
        pivot = array[0]
        for x in array:
            if x < pivot:
```

```
            less.append(x)
        if x == pivot:
            equal.append(x)
        if x > pivot:
            greater.append(x)

    # Don't forget to return something!
    return sort(less)+ equal +sort(greater)  # Just use the + operator to join lists
  # Note that you want equal ^^^^^ not pivot
  else:  # You need to hande the part at the end of the recursion - when you only have one element in your array, just return the array.
    return array
```

3] **Linear Search Algorithm:**
Step 1: Create a function that will accept the data list, the length of the list and the key element
Step 2: If an element present in the given list matches the key element, return the corresponding index number
Step 3: If the element is not found, return -1

Linear Search Program:

```
def lin_search(myarray, n, key):

    for x in range(0, n):
        if (myarray[x] == key):
            return x
    return -1

myarray = [ 12, 1, 34, 17]
key = 17
n = len(myarray)
matched = lin_search(myarray, n, key)
if(matched == -1):
    print("Key is not present")
else:
    print("Key is present in the given list at index", matched)
```

OUTPUT: Key is present in the given list at index 3

4]Binary Search: good if the array is sorted and for small arrays.

Binary Search is used to search for some given element in a sorted array by making use of the Decrease and Conquer Algorithm. Here, the key is looked for by first comparing it with the middle element and then dividing the array into half. The left half is searched if the element to be searched for is smaller than the middle element and vice versa. The appropriate sub-arrays are again divided into half and the process is repeated again.
For example, if you have a sorted list of 8 elements, the key element will be compared with the element present at the middle or 7/ 2 = 3.5 (7 is the index value of the last element) and the whole number value is taken for comparison. So, the key element will be compared with the value present at index number 3 and if the given value is smaller, the same process is repeated towards the left side of the middle element and vice versa.

Binary Search Algorithm:
Step 1: Compare the key with the middle element
Step 2: If matched, return the middle index value
Step 3: If the key element is greater than the middle element, search for the key element towards the right of the middle element, else search to the left of it

Binary Search Program:

```
1  def bin_search(mylist,key):
2      l = 0
3      r = len(mylist)-1
4      matched = False
5      while( l<=r and not matched):
6          middle = (l + r)//2
7          if mylist[middle] == key :
8              matched = True
9          else:
10             if key < mylist[middle]:
11                 r = middle - 1
12             else:
13                 l = middle + 1
14     return matched
15
16
```

4)linked lists:

From the collections import the node and linked list class:

```
class ListNode:
    def __init__(self, data):
        "constructor to initiate this object"

        # store data
        self.data = data

        # store reference (next item)
        self.next = None
        return

    def has_value(self, value):
        "method to compare the value with the node data"
        if self.data == value:
            return True
        else:
            return False
```

Listing 2: Instantiation of nodes

```
node1 = ListNode(15)
node2 = ListNode(8.2)
node3 = ListNode("Berlin")
```

Listing 3: The SingleLinkedList class (part one)

```
class SingleLinkedList:
    def __init__(self):
        "constructor to initiate this object"

        self.head = None
        self.tail = None
        return
```

Listing 4: The SingleLinkedList class (part two)

```
    def add_list_item(self, item):
        "add an item at the end of the list"

        if not isinstance(item, ListNode):
            item = ListNode(item)

        if self.head is None:
            self.head = item
        else:
```

```
    self.tail.next = item
self.tail = item
return
```