Ex No: 01 Date: 12-03-2023

Q1: Write a Python code that takes as input a value n, and generates a list of n unique random values. You may use this code to generate the numbers to be given as input for the searching and sorting code for the questions below.

Code:

```
import random as r
arr=[]
n  = int(input("Enter the number of element : "))
a=0
while(a!=n):
    r1 = r.randint(0,100)
    if r1 not in arr:
        arr.append(r1)
        a+=1
print(arr)
```

Output:

```
┌─[cyber@parrot]─[~/code/Design_and_Analysis_of_Algorithms/Exercise_01]
└──╼ $python3 EX_1.py
Enter the number of element : 20
[7, 14, 1, 36, 15, 56, 87, 59, 68, 0, 71, 21, 48, 35, 53, 76, 100, 40, 20, 44]
┌─[cyber@parrot]─[~/code/Design_and_Analysis_of_Algorithms/Exercise_01]
└──╼ $
```

Q2: Implement insertion sort, shell sort and radix exchange sort, and test the code for arrays of the following size:

- 10
- 100
- 1000
- 10000
- 100000

Code:

```
import random as r
import matplotlib.pyplot as plt

def insertionSort(arr):
        for i in range(1, len(arr)):

                key = arr[i]
                j = i-1
                while j >= 0 and key < arr[j] :
                                arr[j + 1] = arr[j]
```

```
                j -= 1
        arr[j + 1] = key


def shellSort(arr, n):
    gap=n//2
    while gap>0:
        j=gap
        while j<n:
            i=j-gap
            while i>=0:
                if arr[i+gap]>arr[i]:

                    break
                else:
                    arr[i+gap],arr[i]=arr[i],arr[i+gap]

                i=i-gap
            j+=1
        gap=gap//2

def countingSort(array, place):
    size = len(array)
    output = [0] * size
    count = [0] * 10

    for i in range(0, size):
        index = array[i] // place
        count[index % 10] += 1

    for i in range(1, 10):
        count[i] += count[i - 1]

    i = size - 1
    while i >= 0:
        index = array[i] // place
        output[count[index % 10] - 1] = array[i]
        count[index % 10] -= 1
        i -= 1

    for i in range(0, size):
        array[i] = output[i]
def radixSort(array):
    max_element = max(array)
    place = 1
    while max_element // place > 0:
        countingSort(array, place)
        place *= 10


x=[]
y=[]
```

```
n  = int(input("Enter the number of element : "))
a=0
while(a!=n):
    r1 = r.randint(0,100)
    if r1 not in y:
        y.append(r1)
        x.append(a)
        a+=1

insertion_sort_list = y
sell_sort_list = y
radex_sort_list = y

insertionSort(insertion_sort_list)
shellSort(sell_sort_list,n)
radixSort(radex_sort_list)

print("insertion Sort : ",insertion_sort_list)
print("Sell Sort : ",sell_sort_list)
print("radex Sort : ",radex_sort_list)
```

Output:

```
┌──[✗]─[cyber@parrot]─[~/code/Design_and_Analysis_of_Algorithms/Exercise_01]
└─$python3 EX_2.py
Enter the number of element : 13
insertion Sort :  [1, 3, 4, 17, 23, 27, 32, 33, 45, 46, 78, 84, 91]
Sell Sort :  [1, 3, 4, 17, 23, 27, 32, 33, 45, 46, 78, 84, 91]
radex Sort :  [1, 3, 4, 17, 23, 27, 32, 33, 45, 46, 78, 84, 91]
```

Q3: Implement insertion sort, shell sort and radix exchange sort for an array of size 10000 when the input array is:

- Sorted in ascending order
- Sorted in descending order
- Not sorted

Code:

```
import random as r
import matplotlib.pyplot as plt

def insertionSort(arr):
        for i in range(1, len(arr)):

                key = arr[i]
                j = i-1
                while j >= 0 and key < arr[j] :
                                arr[j + 1] = arr[j]
                                j -= 1
                arr[j + 1] = key


def shellSort(arr, n):
        gap=n//2
        while gap>0:
```

```
                j=gap
                while j<n:
                        i=j-gap
                        while i>=0:
                                if arr[i+gap]>arr[i]:

                                        break
                                else:
                                        arr[i+gap],arr[i]=arr[i],arr[i+gap]

                                i=i-gap
                        j+=1
                gap=gap//2

def countingSort(array, place):
    size = len(array)
    output = [0] * size
    count = [0] * 10

    for i in range(0, size):
        index = array[i] // place
        count[index % 10] += 1

    for i in range(1, 10):
        count[i] += count[i - 1]

    i = size - 1
    while i >= 0:
        index = array[i] // place
        output[count[index % 10] - 1] = array[i]
        count[index % 10] -= 1
        i -= 1

    for i in range(0, size):
        array[i] = output[i]
def radixSort(array):
    max_element = max(array)
    place = 1
    while max_element // place > 0:
        countingSort(array, place)
        place *= 10


print("1. Sorted in ascending order\n2.Sorted in descending order\n3. Not sorted")
opt = int(input("Enter your Option : "))


x=[]
y=[]
n  = int(input("Enter the number of element : "))
a=0
while(a!=n):
```

```
    r1 = r.randint(0,100)
    if r1 not in y:
        y.append(r1)
        x.append(a)
        a+=1

insertion_sort_list = y
sell_sort_list = y
radex_sort_list = y

if(opt == 1):
    insertionSort(insertion_sort_list)
    shellSort(sell_sort_list,n)
    radixSort(radex_sort_list)
elif(opt == 2):
    insertionSort(insertion_sort_list)
    shellSort(sell_sort_list,n)
    radixSort(radex_sort_list)
    insertion_sort_list.reverse()
    sell_sort_list.reverse()
    radex_sort_list.reverse()
print("insertion Sort : ",insertion_sort_list)
print("Sell Sort : ",sell_sort_list)
print("radex Sort : ",radex_sort_list)
```

Output:

```
1. Sorted in ascending order
2.Sorted in descending order
3. Not sorted
Enter your Option : 2
Enter the number of element : 14
insertion Sort :  [96, 81, 78, 65, 54, 40, 32, 28, 26, 24, 17, 12, 8, 6]
Sell Sort :  [96, 81, 78, 65, 54, 40, 32, 28, 26, 24, 17, 12, 8, 6]
radex Sort :  [96, 81, 78, 65, 54, 40, 32, 28, 26, 24, 17, 12, 8, 6]
```

Q4: Plot the graphs for both the above implementations with n on the x-axis and time of execution in milliseconds on the y-axis. You may use standard Python packages for plotting the graph.

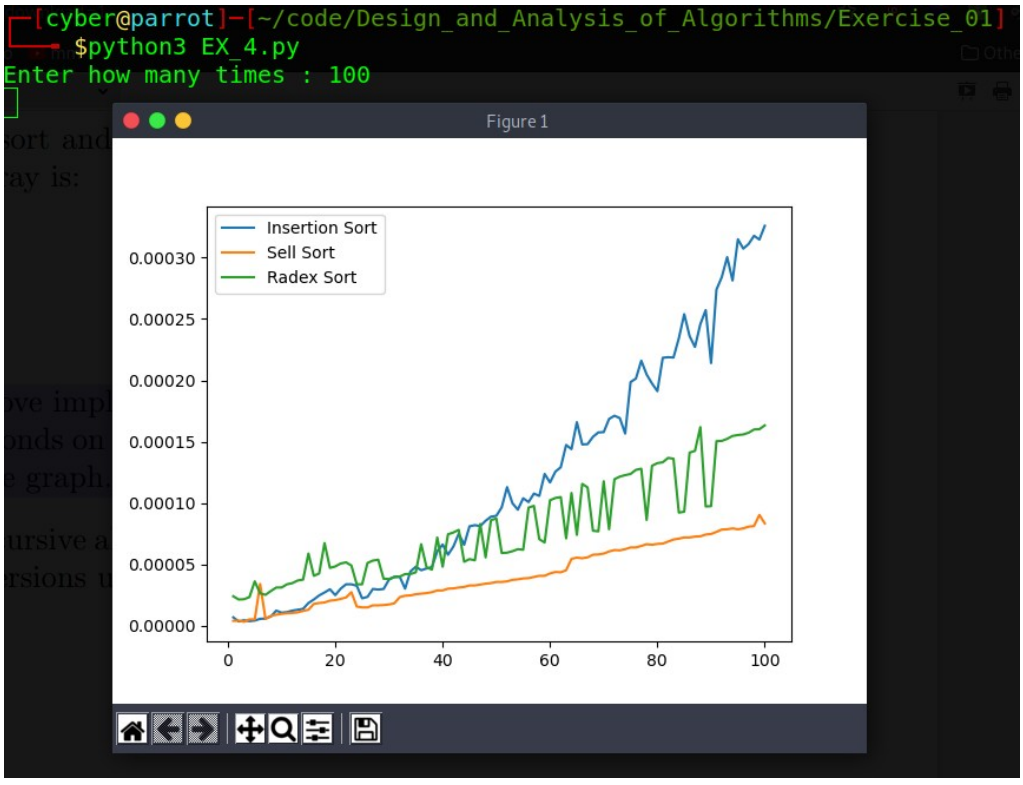
Code:

```
import random as r
import matplotlib.pyplot as plt
import time as t
def insertionSort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >= 0 and key < arr[j] :
            arr[j + 1] = arr[j]
            j -= 1
```

```
        arr[j + 1] = key
def shellSort(array, n):
    interval = n // 2
    while interval > 0:
        for i in range(interval, n):
            temp = array[i]
            j = i
            while j >= interval and array[j - interval] > temp:
                array[j] = array[j - interval]
                j -= interval

            array[j] = temp
        interval //= 2
def countingSort(array, place):
    size = len(array)
    output = [0] * size
    count = [0] * 10

    for i in range(0, size):
        index = array[i] // place
        count[index % 10] += 1

    for i in range(1, 10):
        count[i] += count[i - 1]

    i = size - 1
    while i >= 0:
        index = array[i] // place
        output[count[index % 10] - 1] = array[i]
        count[index % 10] -= 1
        i -= 1

    for i in range(0, size):
        array[i] = output[i]


def radixSort(array):
    max_element = max(array)
    place = 1
    while max_element // place > 0:
        countingSort(array, place)
        place *= 10

num = int(input("Enter how many times : "))
final_y_insertion=[]
final_y_sell=[]
final_y_radex=[]

x=[ a for a in range(1,num+1)]
for i in range(1,num+1):
    y=[]
    a=0
```

```
    while(a!=i):
        r1 = r.randint(0,100)
        if r1 not in y:
            y.append(r1)
            a+=1
    insertion_sort_list = y
    sell_sort_list = y
    radex_sort_list = y

    start = t.time()
    insertionSort(insertion_sort_list)
    end = t.time()
    final_y_insertion.append(end-start)

    start = t.time()
    shellSort(sell_sort_list,len(y))
    end = t.time()
    final_y_sell.append(end-start)

    start = t.time()
    radixSort(radex_sort_list)
    end = t.time()
    final_y_radex.append(end-start)

plt.plot(x,final_y_insertion,label="Insertion Sort")
plt.plot(x,final_y_sell,label="Sell Sort")
plt.plot(x,final_y_radex,label="Radex Sort")
plt.legend()
plt.show()
```

Output:



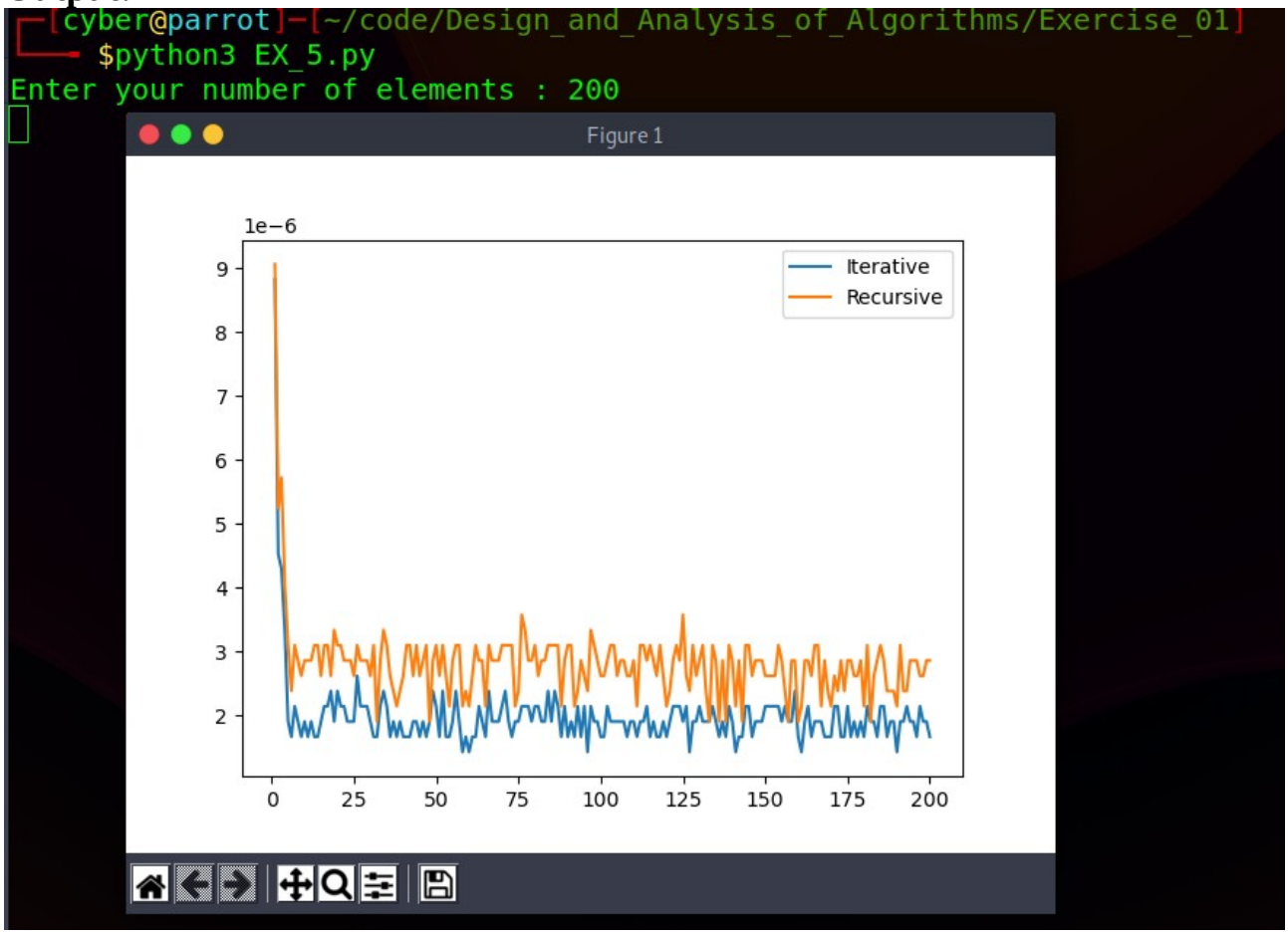
Q5: Implement recursive and non-recursive algorithms for binary search. Compare the performance of both versions using an array of size 10000.

Code:

```
import random
import matplotlib.pyplot as plt
import time as t


def binary_search_I(arr, x):
    low = 0
    high = len(arr) - 1
    mid = 0
    while low <= high:
        mid = (high + low) // 2
        if arr[mid] < x:
            low = mid + 1
        elif arr[mid] > x:
            high = mid - 1
        else:
            return mid
    return -1

def binary_search_R(arr, low, high, x):
    if high >= low:
        mid = (high + low) // 2
        if arr[mid] == x:
            return mid
        elif arr[mid] > x:
            return binary_search_R(arr, low, mid - 1, x)
        else:
            return binary_search_R(arr, mid + 1, high, x)
    else:
        return -1
n = int(input("Enter your number of elements : "))
final_y_it=[]
final_y_re=[]
final_x = [a for a in range(1,n+1)]
for i in range(1,n+1):
    arr=[]
    a=0
    while(n!=a):
        r = random.randint(0,1000)
        if r not in arr:
            arr.append(r)
            a+=1
    find_ele = arr[2]

    start = t.time()
    binary_search_I(arr, find_ele)
    end = t.time()
    final_y_it.append(end-start)
```

```
    start = t.time()
    binary_search_R(arr,  0, len(arr)-1, find_ele)
    end = t.time()
    final_y_re.append(end-start)

plt.plot(final_x,final_y_it,label="Iterative")
plt.plot(final_x,final_y_re, label="Recursive")
plt.legend()
plt.show()
```

Output:



----------***----------