

Institutional Affiliation

Professor's Name

Student's Name

Date

**EXERCISE****1.**

- a) False
- b) True
- c) False
- d) True
- e) False

**2.**

- a) 8 comparisons, not found
- b) 6 comparison
- c) 1 comparison
- d) 8 comparisons, not found

**3. ,**

- a) Sequential search searches for elements sequentially up to search elements is less than element in the list, but does not compare the elements from the list. Compare that search element whether it is greater than each element in the list until it reaches less than or no element in the list. Compare that search element with that element in the list if equal, element is found, if not equal, exits the operation.

An example is shown below;

```

#include <iostream>
using namespace std;

int binarySearch(int[], int, int, int);

int main()
{
    int num[11] = {5,12,17,35,46,65,78,85,93,110,115};
    int search_num, loc=-1;

    cout<<"Enter the element that you want to search: ";
    cin>>search_num;

    loc = binarySearch(num, 0, 10, search_num);

    if(loc != -1)
    {
        cout<<search_num<<" found in the array at the location: "<<loc;
    }
    else
    {
        cout<<"Element not found";
    }
    return 0;
}

int binarySearch(int a[], int first, int last, int search_num)
{
    int middle;
    if(last >= first)
    {
        middle = (first + last)/2;
        //Checking if the element is present at middle loc
        if(a[middle] == search_num)
        {
            return middle;
        }

        //Checking if the search element is present in greater half
        else if(a[middle] < search_num)
        {
            return binarySearch(a,middle+1,last,search_num);
        }

        //Checking if the search element is present in lower half
        else
        {
            return binarySearch(a,first,middle-1,search_num);
        }
    }
    return -1;
}

```

b) „

- i. Element found 35. The key comparisons are 3
- ii. Element found 60. The key comparisons are 5
- iii. Element found 78. The key comparisons are 6
- iv. Element not found 120. The key comparisons are 11

4. ..

- a) Comparisons 2
- b) Comparisons 4
- c) Comparisons 9
- d) Comparisons 11

## 5. Bubble Sort

```
#include<iostream>
using namespace std;
int main ()
{
    int i,j,temp,pass=0;
    int a[8] = {26,45,17,65,33,55,12,18};
    cout <<"Unsorted Input list ...\n";
    for(i = 0; i<8; i++) {
        cout <<a[i]<<"\t";
    }
    cout<<endl;
    for(i = 0; i<8; i++) {
        for(j = i+1; j<8; j++)
        {
            if(a[j] < a[i]) {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
        pass++;
    }
    cout <<"\nSorted Element List ...\n";
    for(i = 0; i<8; i++) {
        cout <<a[i]<<"\t";
    }
    cout<<"\n\nNumber of passes taken to sort the list:"<<pass<<endl;
    return 0;
}
```

```

Unsorted Input list ...
26      45      17      65      33      55      12      18

Sorted Element List ...
12      17      18      26      33      45      55      65

```

## 6. Selection Sort Algorithm

```

#include<iostream>

using namespace std;
int findSmallest (int[],int);
int main ()
{
    int myarray[10] = {36,55,17,35,63,85,12,48,3,66};
    int pos,temp,pass=0;
    cout<<"\n The list of elements to be Sorted\n";
    for(int i=0;i<10;i++)
    {
        cout<<myarray[i]<<"\t";
    }
    for(int i=0;i<10;i++)
    {
        pos = findSmallest (myarray,i);
        temp = myarray[i];
        myarray[i]=myarray[pos];
        myarray[pos] = temp;
        pass++;
    }
    cout<<"\n\n Sorted list of elements is\n";
    for(int i=0;i<10;i++)
    {
        cout<<myarray[i]<<"\t";
    }
    cout<<"\n\n Number of passes required to sort the array: "<<pass;
    return 0;
}

int findSmallest(int myarray[],int i)
{
    int ele_small,position,j;
    ele_small = myarray[i];
    position = i;
    for(j=i+1;j<10;j++)
    {
        if(myarray[j]<ele_small)
        {
            ele_small = myarray[j];
            position=j;
        }
    }
    return position;
}

```

}

```

The list of elements to be Sorted
36      55      17      35      63      85      12      48      3      66

Sorted list of elements is
3       12      17      35      36      48      55      63      66      85

```

7. ..

One comparison

8. .

One Comparison

9. ..

10 18 21 25 28 12 30 71 32 58 15

10. ..

Number of Key Comparisons is 25

11. ..

### **Bubble Sort**

List of elements n=10, 000

$n(n-1) / 2$

$\Rightarrow \{10000(10000-1) \} / 2$

$\Rightarrow \{10000(9999) \} / 2$

$\Rightarrow \{10000 \times 9999 \} / 2$

$\Rightarrow 49,995,000$  comparisons

### **Insertion Sort**

n=10000

$(n^2+3n-4) / 4$  is the formula

$\{(10000)^2 + 3(10000) - 4\} / 4$

$\Rightarrow 100000000 + 30000 - 4$

$\Rightarrow 100029996$

$\Rightarrow 100029996 / 4$

=>25,007,499 Comparisons

### Selection Sort

Similar to bubble sort

$n(n-1)/2$  is formula

- ⇒  $\{10000(10000-1)\}/2$
- ⇒  $\{10000(9999)\}/2$
- ⇒  $\{10000 \times 9999\}/2$
- ⇒ 49,995,000 comparisons

## 12. .. Iterations using Bubble sort

```
a)
    // Function of Bubble Sort.
    void BubbleSort (int arr[], int n)
    {
        int i, j;
        for (i = 0; i < n; ++i)
        {
            for (j = 0; j < n-i-1; ++j)
            {
                // Comparison in iteration. No swaping.
                if (arr[j] > arr[j+1])
                {
                    arr[j] = arr[j]+arr[j+1];
                    arr[j+1] = arr[j]-arr[j + 1];
                    arr[j] = arr[j]-arr[j + 1];
                }
            }
        }
    }
```

b) ..  
16 Comparisons are needed

## 13. Binary Search

it takes 12 maximum number of comparisons in binary search to find the element

## 14. . Rearranging in Ascending Order

```

#include <iostream>
using namespace std;
int Ascending();

int main(){
    int num[100],n;
    int i,j,man;
    cout<<"\nHow many Elements are you Arranging?: "<<endl<<endl;
    cin>>n;
    cout<<"\nEnter The\tn"<<n<<"\tDescending Elements:"<<endl;
    for(i=0;i<n;i++){

        cout<<"\nEnter:"<<endl;
        cin>>num[i];
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(num[i]<num[j]){
                man=num[i];
                num[i]=num[j];
                num[j]=man;
            }
        }
    }
    cout<<"\nAscending output is: "<<endl;
    for(i=0;i<n;i++){
        cout<<" "<<num[i]<<endl;;
    }
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(num[i]>num[j]){
                man=num[i];
                num[i]=num[j];
                num[j]=man;
            }
        }
    }

    return 0;
}

```

```
How many Elements are you Arranging?:
4
Enter The      4      Descending Elements:
Enter:
90
Enter:
76
Enter:
56
Enter:
34
Ascending output is:
34
56
76
90
Process returned 0   execution time : 13.793 s
Press any key to continue.
```

## 15. Vector

*#include <vector>*

*Or*

*#include "Vector.hpp"*

## 16. .

**a)**

It creates a vector of capacity 50

**b)** Creates a vector of capacity 0

17. Elements will be stored as  $2*i + 2$

$\Rightarrow i = 0..4 \Rightarrow 2, 4, 6, 8, 10$

Hence Output will be 2 4 6 8 10

18.

Output will be Nisha Tony Bobby Peter

19.

a)

```
vector<int> secretList;
```

b)

```
secretList.push_back(56);
secretList.push_back(28);
secretList.push_back(32);
secretList.push_back(96);
secretList.push_back(75);
```

c)

```
for(int i=0 ; i<secretList.size(); ++i)
cout<<secretList[i] <<" ";
```

20. Elements will be stored as  $2*i + 5$

$\Rightarrow i = 0..4 \Rightarrow 5, 7, 9, 11, 13, 15, 17, 19, 21, 23$

Hence Output will be 5 7 9 11 13 15 17 19, 21, 23

21.

```
#include <iostream>
#include <list>
using namespace std;

int main()
{
    // declaration of list container
    list<int> mylist{ 1, 2, 3, 4, 5 };

    // using begin() to print list
    for (auto it = mylist.begin(); it !=
        mylist.end(); ++it)
```

```
    cout << ' ' << *it;
return 0;
```

```
c) #include <iostream>
using namespace std;
int main() {
    int len = sizeof(arr)/sizeof(arr[0]);
    cout << "The length of the array is: " << len;
    return 0;
}
```

## 22. Difference between size and capacity of vector

The size of a vector represents the number of components in the vector while the capacity of a vector represents the maximum number of elements the vector can hold

## PROGRAMMING EXERCISES

Solution to these exercises is in a .zip file

For example, Screenshot of solution for question 19 is here;

The image shows a C++ source file named Q19.cpp and its execution output. The source code defines a program that reads scores from a file, displays them, sorts them, and then shows a frequency table.

```
1  #include <iostream> //include statement(s)
2  #include <iomanip>
3  #include <fstream>
4  using namespace std; //using namespace statement(s)
5  const int MAX_SIZE = 999; //constant variable declaration(s)
6  void getScoresAndSize(int array[], int& arraySize); //void function header(s)
7  void displayScoreArray(int array[], int arraySize);
8  void orderArray(int array[], int array2[], int arraySize);
9  void outputData(int array[], int arraySize);
10 int main()
11 {
12     int scoreArray[MAX_SIZE]; //variable declaration(s)
13     int newScoreArray[MAX_SIZE];
14     // ... (rest of the code is in the output window)
15 }
```

The execution output shows the following steps:

```
16 The Q19 Input.txt file was succesfully opened by the program.
17 The scores from the file were succesfully transferred into an array.
18 The array contains 17 scores.
19
20 The Scores are:
21 55 80 78 92 95 55 78 53 92 65 78 95 85 92 85 95 95
22
23 After reordering the array, The Scores are:
24 53 55 55 65 78 78 78 80 85 85 92 92 92 95 95 95 95
25
26 Score      Count
27 53          1
28 55          2
29 65          1
30 78          3
31 80          1
32 85          2
33 92          3
34 95          4
35
36 Program is exiting...
37
38 Process returned 0   execution time : 0.038 s
39 Press any key to continue.
40
41
42
43
44
```

```
C:\Users\S. Engineer Kioko\Desktop\Jeff\Q19.exe
The Q19 Input.txt file was succesfully opened by the program.
The scores from the file were succesfully transferred into an array.
The array contains 17 scores.

The Scores are:
55 80 78 92 95 55 78 53 92 65 78 95 85 92 85 95 95

After reordering the array, The Scores are:
53 55 55 65 78 78 78 80 85 85 92 92 92 95 95 95 95

Score      Count
53          1
55          2
65          1
78          3
80          1
85          2
92          3
95          4

Program is exiting...

Process returned 0   execution time : 0.135 s
Press any key to continue.
```