Name: Sanket Wakankar

Reg. No.: 2020BIT055

Practical No. 3: Write C++ code to implement concept of

1) Searching Algorithm
2) Sorting Algorithm

LINEAR SEARCH

```cpp
// Sanket Wakankar
#include<iostream>
using namespace std;
void search(int arr[],int m,int t){
    for(int i=0;i<m;i++){
        if(arr[i]==t){
            cout<<"target found at index "<<i<<endl;
            return;
        }
    }
    cout<<"target not found"<<endl;
}
int main(){
    int n;
    cout<<"please enter the no. you want in a array"<<endl;
    cin>>n;
    int arr[n];
    cout<<"please enter the numbers in array"<<endl;
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    int target;
    cout<<"please enter the target"<<endl;
    cin>>target;
    search(arr,n,target);
}
```

OUTPUT:

```
PS C:\Users\DELL\Desktop\DAA Practicals> cd "c:\Users\DELL\Desktop\DAA Practicals\" ; if ($?) { g++ linear.cpp -o linear }
please enter the no. you want in a array
4
please enter the numbers in array
1 3 4 66
please enter the target
4
target found at index 2
PS C:\Users\DELL\Desktop\DAA Practicals>
```

BINARY SEARCH

```cpp
// Sanket Wakankar
#include<iostream>
using namespace std;
int binary_search(int narr[],int t,int c){
    int low = 0;
    int high = c-1;
    while(low<=high){
        int mid = (low + high)/2;
        if(t==narr[mid]){
            return mid;
        }
        else if(narr[mid]<t){
            low = mid+1;
        }
        else if(narr[mid]>t){
            high = mid-1;
        }
    }
    return -1;
}
int main(){
    int narr[] = {1,2,3,4,5};
    int target = 9;
    int ans = binary_search(narr,target,5);
    if(ans==-1){
        cout<<"target not found"<<endl;
    }
    else{
        cout<<"target found at index "<<ans<<endl;
    }
}
```

OUTPUT:

```
PS C:\Users\DELL\Desktop\DAA Practicals> cd "c:\Users\DELL\Desktop\DAA Practicals\" ; if ($?) { g
y_search }
target not found
PS C:\Users\DELL\Desktop\DAA Practicals>
```

JUMP SEARCH

```cpp
// Sanket Wakankar
#include<iostream>
#include<cmath>
```

```
using namespace std;
int jump_Search(int a[], int n, int item) {
   int i = 0;
   int m = sqrt(n);

   while(a[m] <= item && m < n) {

      i = m;
      m += sqrt(n);
      if(m > n - 1)
         return -1;
   }

   for(int x = i; x<m; x++) {
      if(a[x] == item)
         return x;
   }
   return -1;
}

int main() {
   int n, item, loc;
   cout << "\n Enter number of items: ";
   cin >> n;
   int arr[n]; //creating an array of size n
   cout << "\n Enter items: ";

   for(int i = 0; i< n; i++) {
      cin >> arr[i];
   }

   cout << "\n Enter search key to be found in the array: ";
   cin >> item;
   loc = jump_Search(arr, n, item);
   if(loc>=0)
      cout << "\n Item found at location: " << loc;
   else
      cout << "\n Item is not found in the list.";
}
```

OUTPUT:

```
PS C:\Users\DELL\Desktop\DAA Practicals> cd "c:\Users\DELL\Desktop\DAA Practicals\" ; if ($
ch }

 Enter number of items: 4

 Enter items: 12
14
15
17

 Enter search key to be found in the array: 45

 Item is not found in the list.
PS C:\Users\DELL\Desktop\DAA Practicals>
```

BUBBLE SORT

```cpp
// Sanket Wakankar
#include <iostream>
using namespace std;

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int arr[] = {89,23,55,23,78,1,0,100};
    int n = sizeof(arr)/sizeof(arr[0]);
    bubbleSort(arr, n);
    cout << "Sorted array: \n";
    for (int i=0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

OUTPUT:

```
PS C:\Users\DELL\Desktop\DAA Practicals> cd "c:\Users\DELL\Desktop\DAA Practicals\" ; if ($?) { g++ bubble_sort.cpp -o
rt }
Sorted array:
0 1 23 23 55 78 89 100
PS C:\Users\DELL\Desktop\DAA Practicals>
```

QUICK SORT

```cpp
// Sanket Wakankar
#include <iostream>
using namespace std;

int partition(int arr[], int low, int high) {
    int x = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < x) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
    return (i + 1);
}

void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int arr[] = {12,43,11,33,66,32,20,100};
    int n = sizeof(arr) / sizeof(arr[0]);
    quickSort(arr, 0, n - 1);
    cout << "Sorted array: \n";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

OUTPUT:

```
PS C:\Users\DELL\Desktop\DAA Practicals> cd "c:\Users\DELL\Desktop\DAA Practicals\" ; if ($?) {
}
Sorted array:
11 12 20 32 33 43 66 100
PS C:\Users\DELL\Desktop\DAA Practicals>
```

SELECTION SORT

```cpp
// Sanket Wakankar
#include <iostream>
using namespace std;

void selectionSort(int arr[], int n) {
    int i, j, minIndex, temp;
    for (i = 0; i < n-1; i++) {
        minIndex = i;
        for (j = i+1; j < n; j++) {
            if (arr[j] < arr[minIndex])
                minIndex = j;
        }
        temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
}

int main() {
    int arr[] = {32,12,44,2,41,67,76,100};
    int n = sizeof(arr)/sizeof(arr[0]);
    selectionSort(arr, n);
    cout << "Sorted array: \n";
    for (int i=0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

OUTPUT:

```
PS C:\Users\DELL\Desktop\DAA Practicals> cd "c:\Users\DELL\Desktop\DAA Practicals\" ; if ($?)
ection_sort }
Sorted array:
2 12 32 41 44 67 76 100
PS C:\Users\DELL\Desktop\DAA Practicals>
```