Data Structure Lab

Assignment No-12-13

Name : Sanket Shivaji Jadhav.

Prn: 2020BTECS00005

1. Searching and Sorting Techniques.

Code:

```
 # include <bits/stdc++.h>
using namespace std;
// Searching
// 1.Linear Search
void ls(vector<int>v,int ele){
    for(auto a:v){
        if(a==ele){
            cout<<"Element Present In a Vector.\n";
            return;}
    }
    cout<<"Element Not Present In a Vector\n";
}

// 2.Binary Search
void binarysearch(vector<int>v,int ele){
int i=0,j=v.size()-1;
    while(i<=j){
        int mid=i+(j-i)/2;
        if(v[mid]==ele){
            cout<<"Element Present In a Vector.\n";
            return;
        }
        else if(v[mid]>ele){
            j=mid-1;
        }
        else{
```

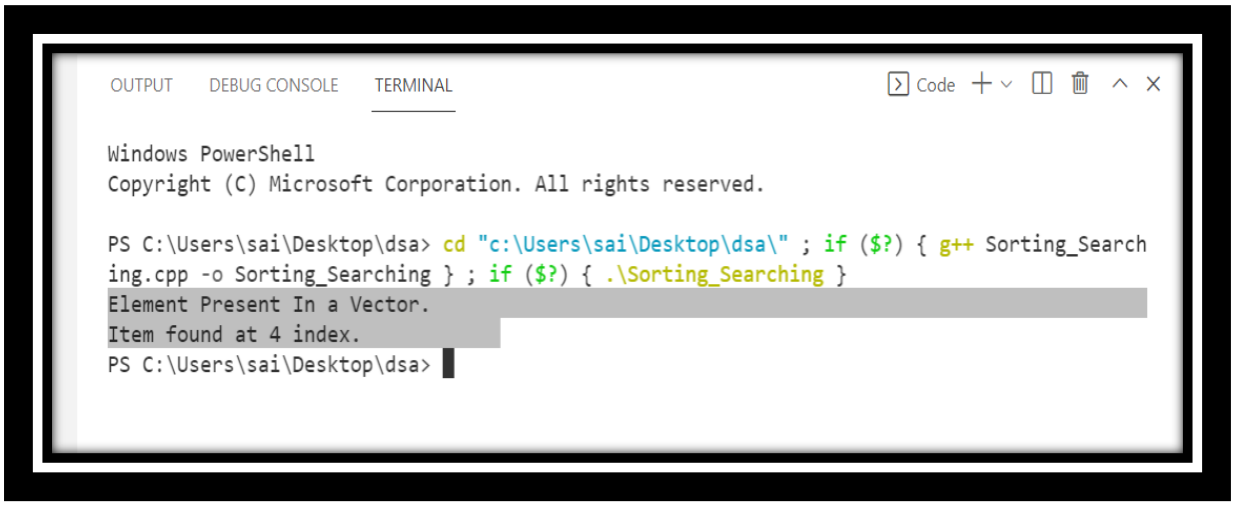
```
            i=mid+1;
        }
    }
    cout<<"Element Not Present In a Vector\n";
}

// 3.Fibonacci Search
void FibonacciSearch(int *a, int start, int end, int *fab, int
index, int item)
{
    int i, mid;
    mid = start+fab[index-2];

    if(item == a[mid]){
        cout<<"Item found at "<<mid<<" index.";
        return;
    }

    else if(item == a[start]){
        cout<<"Item found at "<<start<<" index.";
        return;
    }

    else if(item == a[end]){
        cout<<"Item found at "<<end<<" index.";
        return;}

    else if(mid == start || mid == end){
        cout<<"\nElement not found";
        return;
    }

    else if(item > a[mid])
        FibonacciSearch(a, mid, end, fab, index-1, item);
    else
        FibonacciSearch(a, start, mid, fab, index-2, item);
}

// Main Function
```

```
int main(){
    vector<int>v={2,3,4,5,6,7,9};
    binarysearch(v,5);

    int i,fab[20];
    int
a[20]={1,9,18,24,27,35,38,41,49,53,55,66,67,72,75,77,81,89,90,97}
;

    fab[0] = 0;
    fab[1] = 1;
    i = 1;
    while(fab[i] < 20)
    {
        i++;
        fab[i] = fab[i-1]+fab[i-2];
    }

    FibonacciSearch(a, 0, 19, fab, i, 27);
    return 0;
}
```

INPUT:

```
int
a[20]={1,9,18,24,27,35,38,41,49,53,55,66,67,72,75,77,81,8
9,90,97};
```

OUTPUT:



2. Sorting .

1.Merge Sort :

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

void merge(int arr[], int p, int q, int r) {

  int n1 = q - p + 1;
  int n2 = r - q;

  int L[n1], M[n2];

  for (int i = 0; i < n1; i++)
    L[i] = arr[p + i];
  for (int j = 0; j < n2; j++)
    M[j] = arr[q + 1 + j];

 int i, j, k;
  i = 0;
  j = 0;
  k = p;

  while (i < n1 && j < n2) {
```

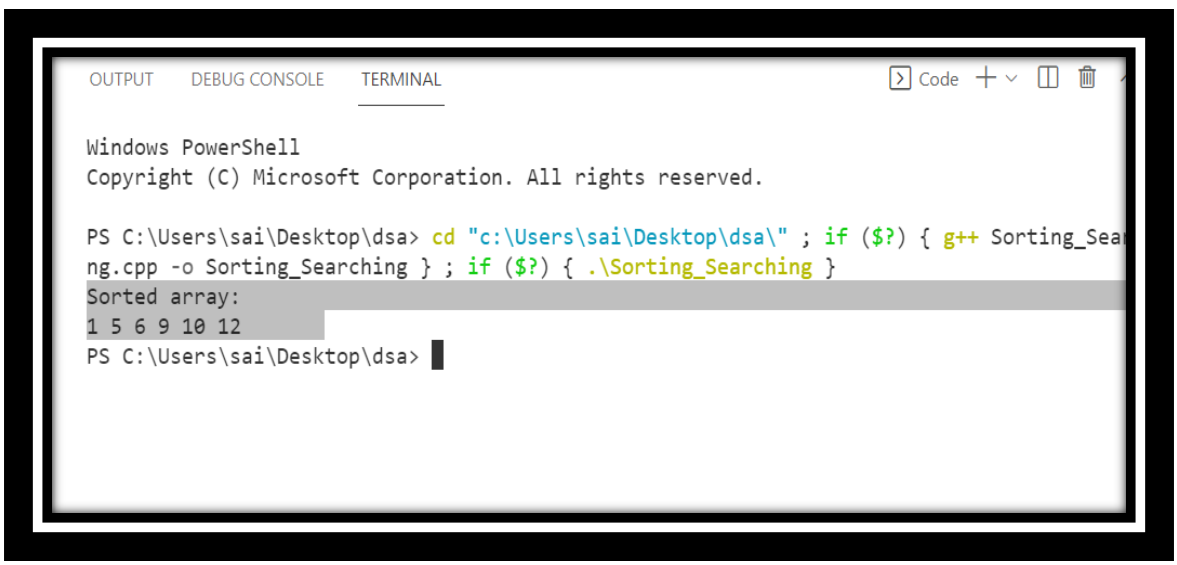
```
    if (L[i] <= M[j]) {
      arr[k] = L[i];
      i++;
    } else {
      arr[k] = M[j];
      j++;
    }
    k++;
  }

  while (i < n1) {
    arr[k] = L[i];
    i++;
    k++;
  }

  while (j < n2) {
    arr[k] = M[j];
    j++;
    k++;
  }
}


void mergeSort(int arr[], int l, int r) {
  if (l < r) {

    int m = l + (r - l) / 2;

    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    merge(arr, l, m, r);
  }
}

void printArray(int arr[], int size) {
  for (int i = 0; i < size; i++)
    cout << arr[i] << " ";
```

```
  cout << endl;
}


int main() {
  int arr[] = {6, 5, 12, 10, 9, 1};
  int size = sizeof(arr) / sizeof(arr[0]);

  mergeSort(arr, 0, size - 1);

  cout << "Sorted array: \n";
  printArray(arr, size);
  return 0;
}
```

INPUT:

```
int arr[] = {6, 5, 12, 10, 9, 1};
```

OUTPUT:



2.Quick Sort :

```
#include <bits/stdc++.h>
using namespace std;
```

```
void swap(int* a, int* b){
    int t = *a;
    *a = *b;
    *b = t;
}

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

   for (int j = low; j <= high - 1; j++)
    {
        if (arr[j] < pivot)
        {
            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);
    }
}

void printArray(int arr[], int size)
{
    int i;
    for (i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
```

```
}

// Main Function
int main()
{
    int arr[] = {1,18,23,71, 16, 90};
    int n = sizeof(arr) / sizeof(arr[0]);
    quickSort(arr, 0, n - 1);
    cout << "Sorted array: \n";
    printArray(arr, n);
    return 0;
}
```

INPUT:

```
int arr[] = {1,18,23,71, 16, 90};
```

OUTPUT: