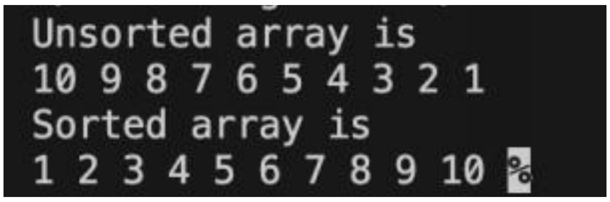
Arnav Goel 1024030530 2C35

DSA Assignment 7

Q1 Bubble Sort

```cpp
// here we will see the code of bubble sort

#include <iostream>
using namespace std;

int main(){
    int arr [] = {10,9,8,7,6,5,4,3,2,1};
    cout<<"Unsorted array is "<<endl;
    for(int i=0;i<10;i++){
        cout<<arr [i]<<' ';
    }
    cout<<endl;
    for(int i=0;i<9;i++){
        bool variable = false;
        for(int j=0;j<10-i-1;j++){
            if(arr [j+1]<arr [j]){
                swap(arr [j],arr [j+1]);
                variable = true;
            }
        }
        if( !variable){
            break;
        }
    }
    cout<<"Sorted array is "<<endl;
    for(int i=0;i<10;i++){
        cout<<arr [i]<<' ';
    }
}
```

Output:

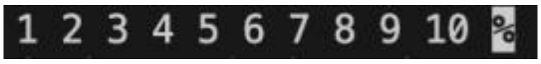


```
Unsorted array is
10 9 8 7 6 5 4 3 2 1
Sorted array is
1 2 3 4 5 6 7 8 9 10 %
```

Q2 Selection Sort

```
// here we will see the code of selection sort

#include <iostream>
using namespace std;

int main(){
    int arr [] = {10,9,8,7,6,5,4,3,2,1};
    for(int i=0;i<9;i++){
        int min = arr [i];
        int index = i;
        for(int j=i+1;j<10;j++){
            if(arr [j]<min){
                min = arr [j];
                index = j;
            }
        }
        swap(arr [i],arr [index]);
    }
    for(int i=0;i<10;i++){
        cout<<arr [i]<<' ';
    }
}
```

Output:

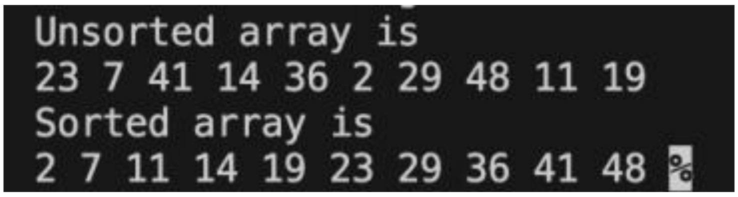


```
1 2 3 4 5 6 7 8 9 10 %
```

Q3 Insertion Sort

```
// here we will see how to perform insertion sort

#include <iostream>
using namespace std;

int main(){
    int a [] = {23, 7, 41, 14, 36, 2, 29, 48, 11, 19};
    int i,j;
    cout<<"Unsorted array is "<<endl;
    for(i=0;i<10;i++){
        cout<<a [i]<<' ';
    }
    for(i=0;i<9;i++){
        int temp = a [i+1];
        for(j=i+1;j>0;j--){
            if(temp<a [j-1]){
                a [j] = a [j-1];
                }
                else break;
            }
            a[j] = temp;
        }
        cout<<endl<<"Sorted array is "<<endl;
        for(i=0;i<10;i++){
            cout<<a[i]<<' ';
        }
}
```

Output:

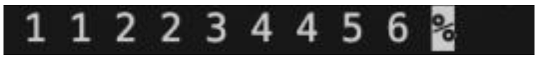


```
Unsorted array is
23 7 41 14 36 2 29 48 11 19
Sorted array is
2 7 11 14 19 23 29 36 41 48 %
```

Q4 Merge Sort

```
// here we will see the code of merge sort
// refer pdf explanation of this code
// time complexity of merge sort is O(nlog(n))

#include <iostream>
using namespace std;

void merge(int arr[], int low, int mid, int high){
    vector<int> temp;
    int left = low; // first array is from [left .. mid]
    int right = mid+1; // second array is from [mid+1 .. high]
    while(left<=mid && right<=high){
        if(arr[left]<arr[right]){
            temp.push_back(arr[left]);
            left++;
        }
        else{
            temp.push_back(arr[right]);
            right++;
        }
    }
    while(left<=mid){
        temp.push_back(arr[left]); // for remaining elements of
left if any are there
        left++;
    }
    while(right<=high){
        temp.push_back(arr[right]); // for remaining elements of
right if any are there
        right++;
    }
    for(int i=low;i<=high;i++){
        arr[i] = temp[i-low];
    }

}

void merge_sort(int arr[], int low, int high){
    if(low>=high)return; // base case
    int mid = (low+high)/2;
    merge_sort(arr,low,mid); // sorts lhs side of the array
    merge_sort(arr,mid+1,high); // sorts rhs side of the array
    merge(arr,low,mid,high); // merges two sorted array into one
sorted array
}

int main(){
    int arr[] = {3,1,2,4,1,5,2,6,4};
    int low = 0;
    int high = 8;
    merge_sort(arr,low,high);
    for(int i=0;i<9;i++){
        cout<<arr[i]<<' ';
    }
}
```

Output:

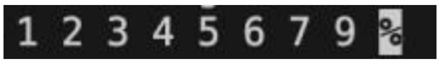


```
1 1 2 2 3 4 4 5 6 %
```

Q5 Quick Sort

```
// here we will see about quick sort algorithm
// it has time complexity of O(nlog(n)) and space complexity is
O(1)
// read the pdf for the explanation of the code

#include <iostream>
using namespace std;

int partition(int arr[], int low, int high){
    int i=low;
    int j=high;
    int pivot = arr[low];
    while(i<j){
        while(arr[i]<=pivot && i<=high-1) i++;
        while(arr[j]>=pivot && j>=low+1) j--;
        if(i<j) swap(arr[i],arr[j]);
    }
    swap(arr[low],arr[j]);
    return j;
}

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

int main(){
    int arr[] = {4,6,2,5,7,9,1,3};
    int low = 0;
    int high = 7;
    quick_sort(arr,low,high);
    for(int i=0;i<8;i++){
        cout<<arr[i]<<' ';
    }
}
```

Output:

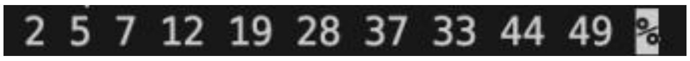


```
1 2 3 4 5 6 7 9 %
```

Q6 Improved Selection Sort

```
// slightly improver version of selection sort

#include <iostream>
using namespace std;

int main() {
    int arr[] = {37,12,49,5,28,44,7,19,33,2};
    for (int i=0;i<5;i++) {
        int min = arr[i];
        int max = arr[9-i];
        int min_index = i;
        int max_index = 9-i;
        for (int j=i+1;j<10-i;j++) {
            if (arr[j]<min){
                min = arr[j];
                min_index = j;
            }
            if (arr[j]>max){
                max = arr[j];
                max_index = j;
            }
        }
        swap(arr[i], arr[min_index]);
        if (max_index == i) { // if we swapped max_index element
where it was supposed to be originally
            max_index = min_index;
        }
        swap(arr[9 - i], arr[max_index]);
    }
    for (int i=0;i<10;i++) {
        cout<<arr[i]<< ' ';
    }
}
```

Output:



```
2 5 7 12 19 28 37 33 44 49 %
```