



Assignment

**Course:** Data Structure

**Year:** 2020



American International University-Bangladesh

Faculty of Science and Technology

Department of Computer Science

CSC 2105: Data Structure (Theory)

**Assignment-for-Final**

1. Write a program to compare the time taken by three different sorting algorithms (Bubble sort, selection sort, insertion sort) for various number of input sizes. That means, three functions for the three sorting algorithms. Additionally, you need to display the count of **comparisons** and **swaps** for all the sorting.
2. Implement the insertion, deletion in one-way (singly) and two-way (doubly) linked list in your preferred programming language (C/ C++/ JAVA).

**Answer:**

**Answer to the question no 1**

**Bubble sort:**

#include <iostream>

using namespace std;

int comp\_no =0, swap\_no = 0;

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void bubbleSort(int arr[], int n)

{

int i, j;

for (i = 0; i < n-1; i++)

{

for (j = 0; j < n-i-1; j++)

{

comp\_no++;

if (arr[j] > arr[j+1])

{

swap\_no++;

swap(&arr[j], &arr[j+1]);

}

}

}

}

void printArray(int arr[], int size)

{

int i;

for (i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

cout << endl;

}

int main()

{

int n;

cout << "Please enter the size of an array: ";

cin >> n;

int arr[n];

cout << "Please enter the elements of the array: ";

for(int i = 0; i < n; i++)

{

cin >> arr[i];

}

cout<<"\nUnsorted array: ";

for(int i = 0; i < n; i++)

{

cout << arr[i] << " ";

}

cout << endl << endl;

bubbleSort(arr, n);

cout<<"Sorted array: ";

printArray(arr, n);

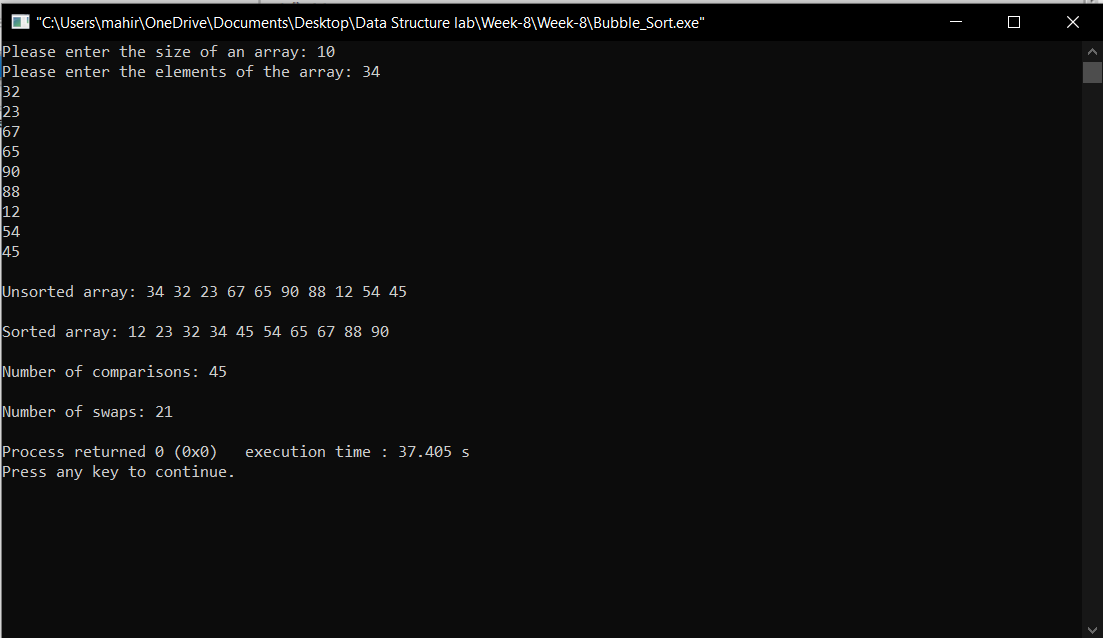
cout << "\nNumber of comparisons: " << comp\_no << endl;

cout << "\nNumber of swaps: " << swap\_no << endl;

return 0;

}

**Output:**



**Selection sort:**

#include<iostream>

using namespace std;

int comp\_no =0, swap\_no = 0;

int main()

{ int n;

cout << "Please enter the size of an array: ";

cin >> n;

int a[n];

cout << "Please enter the elements of the array: ";

for(int i = 0; i < n; i++)

{

cin >> a[i];

}

cout<<"\nUnsorted array: ";

for(int i = 0; i < n; i++)

{

cout << a[i] << " ";

}

cout << endl << endl;

int i,j,loc,temp,min,accept\_swap;

for(i=0;i<n-1;i++)

{

min=a[i];

loc=i;

for(j=i+1;j<n;j++)

{

comp\_no++;

if(min>a[j])

{

min=a[j];

accept\_swap = j;

loc=accept\_swap;

}

}

if (loc == accept\_swap)

{

swap\_no++;

}

temp=a[i];

a[i]=a[loc];

a[loc]=temp;

}

cout<<"\nSorted Elements: ";

for(i=0;i<n;i++)

{ cout<<a[i]<<" ";

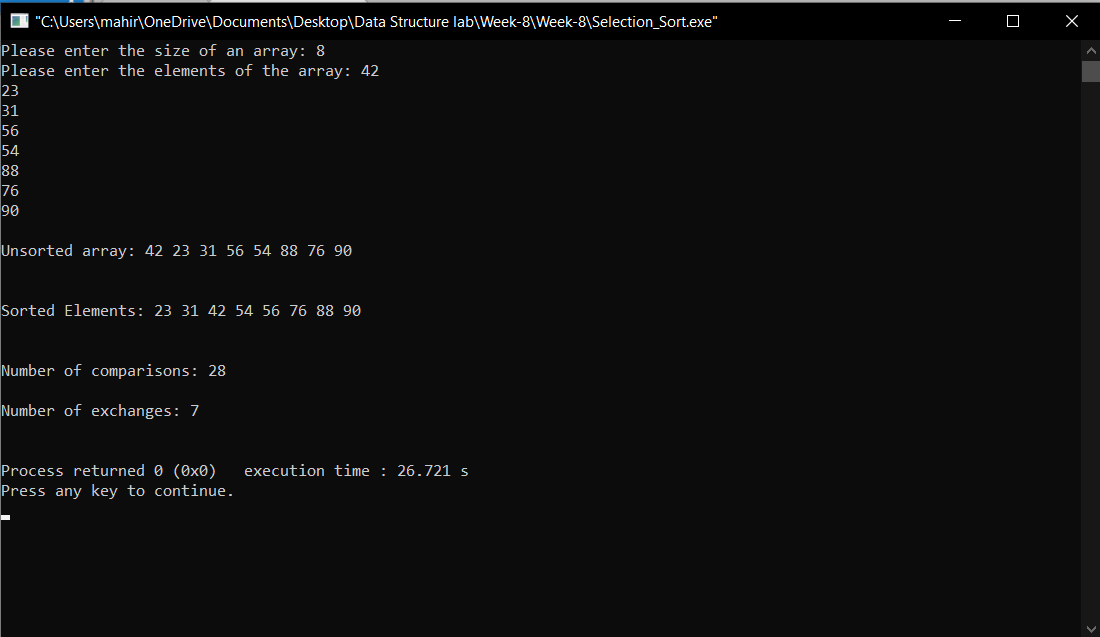
}cout << endl << endl;

cout << "\nNumber of comparisons: " <<comp\_no << endl;

cout << "\nNumber of swap : " <<swap\_no << endl;

return 0;}

**Output:**

****

**Insertion sort:**

#include <iostream>

using namespace std;

int main()

{

int n, i, j, temp, no\_swap=0, comp=0;

cout << "Please enter the size of an array: ";

cin >> n;

int array[n];

cout<< "Please enter the elements of an array: ";

for (i = 0; i < n; i++)

{

cin >> array[i];

}

cout<<"\nUnsorted array: ";

for(int i = 0; i < n; i++)

{

cout << array[i] << " ";

}

cout << endl << endl;

for (i = 1; i < n; i++)

{

j = i;

while (j > 0)

{

comp++;

if(array[j - 1] > array[j])

{

temp = array[j - 1];

array[j - 1] = array[j];

array[j] = temp;

no\_swap++;

}

j--;

}

}

cout<<"Sorted Elements: ";

for(i=0;i<n;i++)

{

cout<<array[i]<<" ";

}

cout << endl << endl;

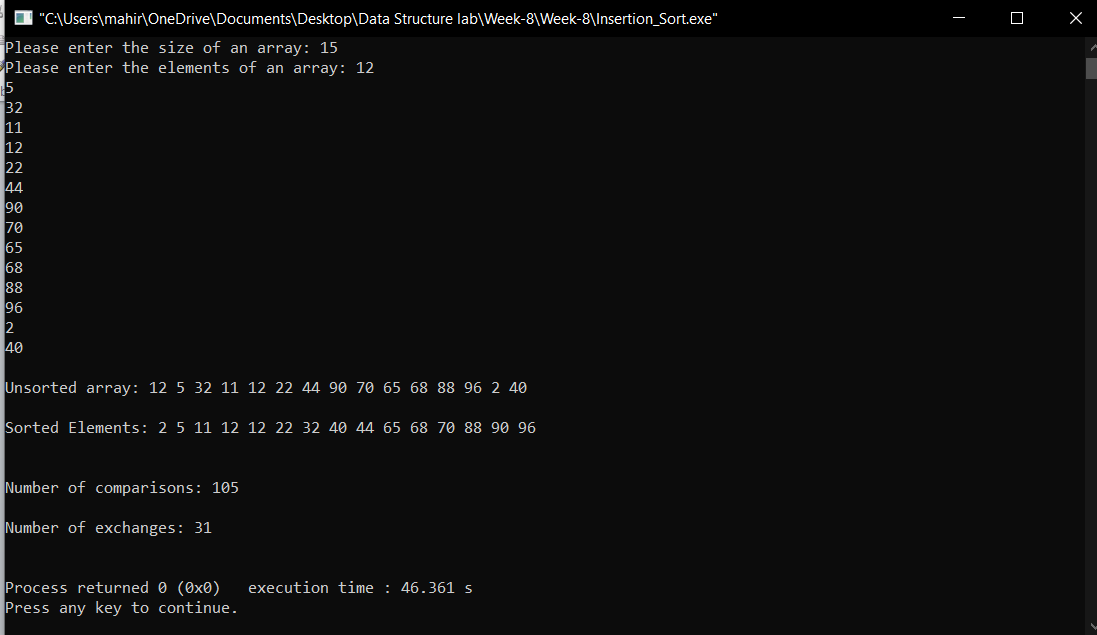
cout << "\nNumber of comparisons: " <<comp << endl;

cout << "\nNumber of exchanges: " <<no\_swap << endl << endl;

return 0;

}

**Output :**



**Answer to the question no 2**

**Singly linked list:**

#include <stdlib.h>

#include <iostream>

using namespace std;

struct Node {

int item;

struct Node\* next;

};

void insertAtBeginning(struct Node\*\* ref, int data)

{

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

new\_node->item = data;

new\_node->next = (\*ref);

(\*ref) = new\_node;

}

void insertAfter(struct Node\* prev\_node, int data) {

if (prev\_node == NULL)

{

cout << "the given previous node cannot be NULL";

return;

}

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

new\_node->item = data;

new\_node->next = prev\_node->next;

prev\_node->next = new\_node;

}

void insertAtEnd(struct Node\*\* ref, int data)

{

struct Node\* new\_node = (struct Node\*)malloc(sizeof(struct Node));

struct Node\* last = \*ref;

new\_node->item = data;

new\_node->next = NULL;

if (\*ref == NULL)

{

\*ref = new\_node;

return;

}

while (last->next != NULL)

last = last->next;

last->next = new\_node;

return;

}

void deleteNode(struct Node\*\* ref, int key)

{

struct Node \*temp = \*ref, \*prev;

if (temp != NULL && temp->item == key)

{

\*ref = temp->next;

free(temp);

return;

}

while (temp != NULL && temp->item != key) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) return;

prev->next = temp->next;

free(temp);

}

void printList(struct Node\* node)

{

while (node != NULL) {

cout << node->item << " ";

node = node->next;

}

}

int main()

{

struct Node\* head = NULL;

insertAtEnd(&head, 1);

insertAtBeginning(&head, 2);

insertAtBeginning(&head, 3);

insertAtEnd(&head, 4);

insertAfter(head->next, 5);

cout << "Linked list: ";

printList(head);

cout << "\nAfter deleting an element: ";

deleteNode(&head, 3);

deleteNode(&head, 1);

printList(head);

cout << endl;

insertAtBeginning(&head, 6);

insertAtBeginning(&head, 7);

cout << "Inserting elements in Linked list: ";

printList(head);

}

**Output:**



**Singly linked list Doubly linked list**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. In Singly linked list, the traversal can be done using the next node link only. | 1. In Doubly linked list, the traversal can be done using the previous node link or the next node link. | |  |
| 1. The Singly linked list occupies less memory than DLL as it has only 2 fields. | 2) The Doubly linked list occupies more memory than SLL as it has 3 fields. | |  |
| 1. Less efficient access to elements. | 3) More efficient access to elements. | |  |
| 1. Singly linked list has nodes with only 2. a data field and next link field. | | 4) Doubly linked list has nodes with a data field, a previous link field and a next link field. | |

**Stay with us:**

**Our Facebook Group:** [**https://www.facebook.com/groups/solutionaiub**](https://www.facebook.com/groups/solutionaiub)

**Our Facebook Page:** [**https://www.facebook.com/AIUBSolution**](https://www.facebook.com/AIUBSolution)

**Our YouTube Channel:**[https://www.youtube.com/c/AIUBSOLUTION-AIUB](https://www.youtube.com/c/AIUBSOLUTION-AIUB?fbclid=IwAR0_YP7mrU_AbR9NEXrMS1iUaYzkGnt7ZBqTuQMM4DwbUCFkldMq1kstIRo)