Name : Bhushan Sharad Tejankar

Roll no. : I30

reg_no. : 2020BIT030

```cpp
// Bhushan Sharad Tejankar
// DAA - Practical 5
// 1.Binary Search
#include <bits/stdc++.h>
using namespace std;
int binarySearch(int a[], int left, int right, int x)
{
  while (left <= right)
  {
    int mid = left + (right - left) / 2;
    if (a[mid] == x)
    {
      return mid;
    }
    else if (a[mid] < x)
    {
      left = mid + 1;
    }
    else
    {
      right = mid - 1;
    }
  }
  return -1;
}

```

```
int main()
{
  int arraySize, a, num;
  int arr[7];
  int output;

  cout << "Enter elments to store in array in ascending order -> ";
  for (int i = 0; i < 7; i++)
  {
    cin >> arr[i];
  }

  cout << "Enter an element for binary search -> ";
  cin >> num;

  output = binarySearch(arr, 0, 6, num);

  if (output == -1)
  {
    cout << "No element found !";
  }
  else
  {
    cout << "The element found at the position " << output;
  }
}
```

```
Enter elments to store in array in ascending order -> 1 3 5 7 9 11 14
Enter an element for binary search -> 11
The element found at the index position 5
PS C:\Users\91830\OneDrive\Desktop>
```

```cpp
// 2.Merge Sort
#include <iostream>
using namespace std;

void merge(int array[], int const left, int const mid,
           int const right)
{
    auto const subArrayOne = mid - left + 1;
    auto const subArrayTwo = right - mid;

    auto *leftArray = new int[subArrayOne],
        *rightArray = new int[subArrayTwo];

    for (auto i = 0; i < subArrayOne; i++)
        leftArray[i] = array[left + i];
    for (auto j = 0; j < subArrayTwo; j++)
        rightArray[j] = array[mid + 1 + j];

    auto indexOfSubArrayOne
        = 0,
        indexOfSubArrayTwo
        = 0;
    int indexOfMergedArray
        = left;

    while (indexOfSubArrayOne < subArrayOne
        && indexOfSubArrayTwo < subArrayTwo) {
```

```
28          if (leftArray[indexOfSubArrayOne]
29              <= rightArray[indexOfSubArrayTwo]) {
30              array[indexOfMergedArray]
31                  = leftArray[indexOfSubArrayOne];
32              indexOfSubArrayOne++;
33          }
34          else {
35              array[indexOfMergedArray]
36                  = rightArray[indexOfSubArrayTwo];
37              indexOfSubArrayTwo++;
38          }
39          indexOfMergedArray++;
40      }
41      while (indexOfSubArrayOne < subArrayOne) {
42          array[indexOfMergedArray]
43              = leftArray[indexOfSubArrayOne];
44          indexOfSubArrayOne++;
45          indexOfMergedArray++;
46      }
47      while (indexOfSubArrayTwo < subArrayTwo) {
48          array[indexOfMergedArray]
49              = rightArray[indexOfSubArrayTwo];
50          indexOfSubArrayTwo++;
51          indexOfMergedArray++;
52      }
53      delete[] leftArray;
54      delete[] rightArray;
```

```
}

void mergeSort(int array[], int const begin, int const end)
{
    if (begin >= end)
        return;

    auto mid = begin + (end - begin) / 2;
    mergeSort(array, begin, mid);
    mergeSort(array, mid + 1, end);
    merge(array, begin, mid, end);
}

void printArray(int A[], int size)
{
    for (auto i = 0; i < size; i++)
        cout << A[i] << " ";
}

int main()
{
    int arr[] = { 12, 11, 13, 5, 6, 7 };
    auto arr_size = sizeof(arr) / sizeof(arr[0]);

    cout << "Given array is \n";
    printArray(arr, arr_size);
```

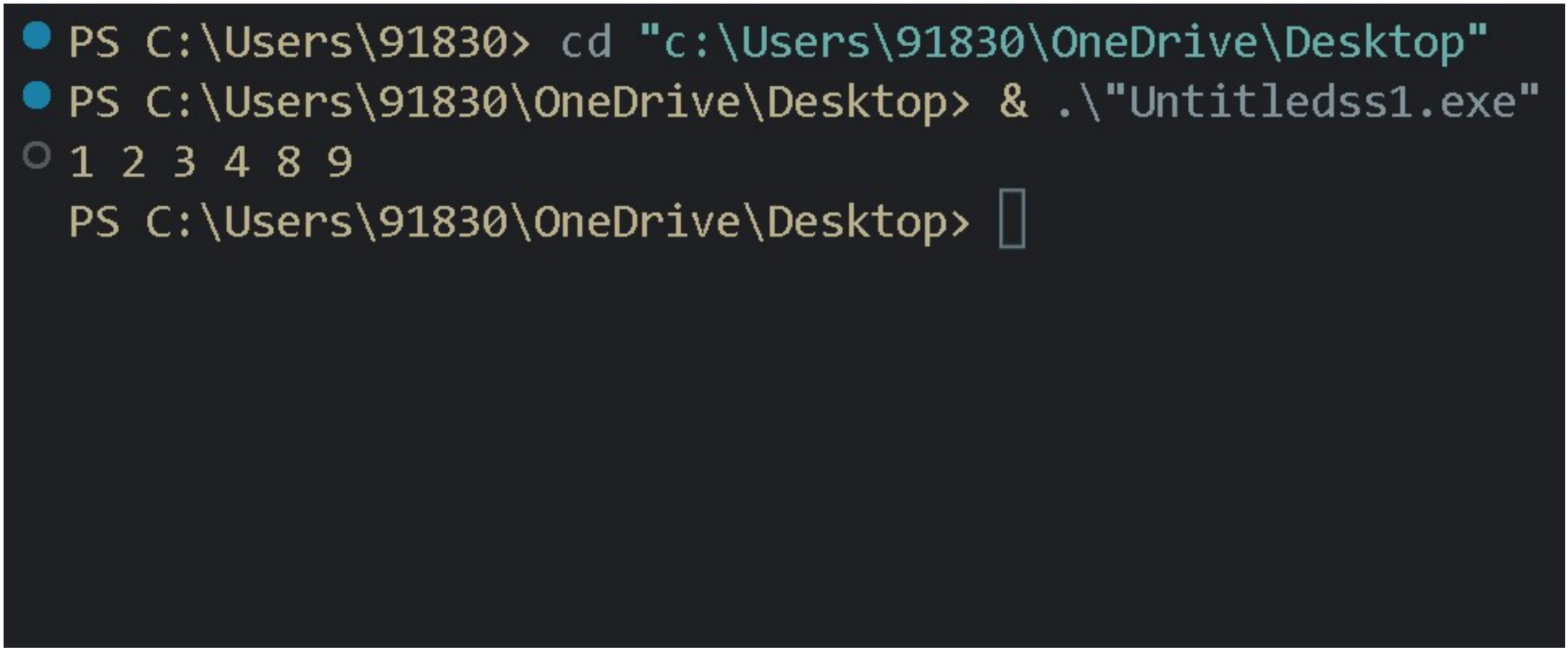
```
81
82      mergeSort(arr, 0, arr_size - 1);
83
84      cout << "\nSorted array is \n";
85      printArray(arr, arr_size);
86      return 0;
87  }
88
89
```

```
Given array is
12 11 13 5 6 7
Sorted array is
5 6 7 11 12 13
PS C:\Users\91830\OneDrive\Desktop>
```

```cpp
// 3.Quick Sort Algorithm
#include <iostream>
using namespace std;

int partition(int arr[], int start, int end)
{

    int pivot = arr[start];

    int count = 0;
    for (int i = start + 1; i <= end; i++) {
        if (arr[i] <= pivot)
            count++;
    }

    // Giving pivot element its correct position
    int pivotIndex = start + count;
    swap(arr[pivotIndex], arr[start]);

    // Sorting left and right parts of the pivot element
    int i = start, j = end;

    while (i < pivotIndex && j > pivotIndex) {

        while (arr[i] <= pivot) {
            i++;
        }

        while (arr[j] > pivot) {
            j--;
```

```
31          }
32
33          if (i < pivotIndex && j > pivotIndex) {
34              swap(arr[i++], arr[j--]);
35          }
36      }
37
38      return pivotIndex;
39  }
40
41  void quickSort(int arr[], int start, int end)
42  {
43
44      // base case
45      if (start >= end)
46          return;
47
48      // partitioning the array
49      int p = partition(arr, start, end);
50
51      // Sorting the left part
52      quickSort(arr, start, p - 1);
53
54      // Sorting the right part
55      quickSort(arr, p + 1, end);
56  }
57
58  int main()
59  {
60
```

```
61      int arr[] = { 9, 3, 4, 2, 1, 8 };
62      int n = 6;
63
64      quickSort(arr, 0, n - 1);
65
66      for (int i = 0; i < n; i++) {
67          cout << arr[i] << " ";
68      }
69
70      return 0;
71  }
72
```


PS C:\Users\91830> cd "c:\Users\91830\OneDrive\Desktop"
PS C:\Users\91830\OneDrive\Desktop> & .\"Untitledss1.exe"
1 2 3 4 8 9
PS C:\Users\91830\OneDrive\Desktop>

```cpp
// 4.Strassen's Matrix multiplication
#include <iostream>
using namespace std;

void multiply(int[5][5], int[5][5], int, int, int);
int display(int[5][5], int, int);

int main()
{

    int a[5][5], b[5][5], r1, c1, r2, c2;
    cout << "\n Enter rows for first matrix: ";
    cin >> r1;
    cout << "\n Enter columns for second matrix: ";
    cin >> c1;

    cout << "\n Enter rows for first matrix: ";
    cin >> r2;
    cout << "\n Enter columns for second matrix: ";
    cin >> c2;

    // To check if columns of first matrix are equal to rows of second matrix

    if (c1 != r2)
        return 0;

    // Storing elements of first matrix.

    cout << "\n Enter elements of first matrix \n";

```

```
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c1; j++)
            cin >> a[i][j];
    }
    // Storing elements of second matrix.
    cout << "\n Enter elements of second matrix\n";

    for (int i = 0; i < r2; i++) {
        for (int j = 0; j < c2; j++)
            cin >> b[i][j];
    }
    display(a, r1, c1);
    display(b, r2, c2);
    //calling the function to multiply a and b. passing number of rows
    //and columns in both of them
    multiply(a, b, r1, c2, c1);
    return 0;
}

void multiply(int a[5][5], int b[5][5], int row, int col, int c1)
{
    int c[5][5];
    //input 0 for all values of c, in order to remove
    //the garbage values assigned earlier
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++)
            c[i][j] = 0;
    }
    //we apply the same formula as above
    for (int i = 0; i < row; i++) {
```

```
61          for (int j = 0; j < col; j++) {
62              for (int k = 0; k < c1; k++) //columns of first matrix || rows of second matrix
63                  c[i][j] += a[i][k] * b[k][j];
64          }
65      }
66      //to display matrix
67      cout << "\n Matrix c after matrix multiplication is:\n";
68      display(c, row, col);
69  }
70
71  int display(int c[5][5], int row, int col)
72  {
73      cout << "\n Matrix is:\n";
74      for (int i = 0; i < row; i++) {
75          for (int j = 0; j < col; j++)
76              cout << c[i][j] << " ";
77          cout << "\n";
78      }
79      return 0;
80  }
81
82
83
84
```

```
PS C:\Users\91830\OneDrive\Desktop> cd "c:\Users\91830\OneDrive\Desktop"
PS C:\Users\91830\OneDrive\Desktop> & .\"Untitledss1.exe"

 Enter rows for first matrix: 2

 Enter columns for second matrix: 2

 Enter rows for first matrix: 2

 Enter columns for second matrix: 2

 Enter elements of first matrix
1 2
3 4

 Enter elements of second matrix
4 3
2 1
```

```
 Matrix is:
1 2
3 4

 Matrix is:
4 3
2 1

 Matrix c after matrix multiplication is:

 Matrix is:
8 5
20 13
PS C:\Users\91830\OneDrive\Desktop>
```


THANK
YOU!