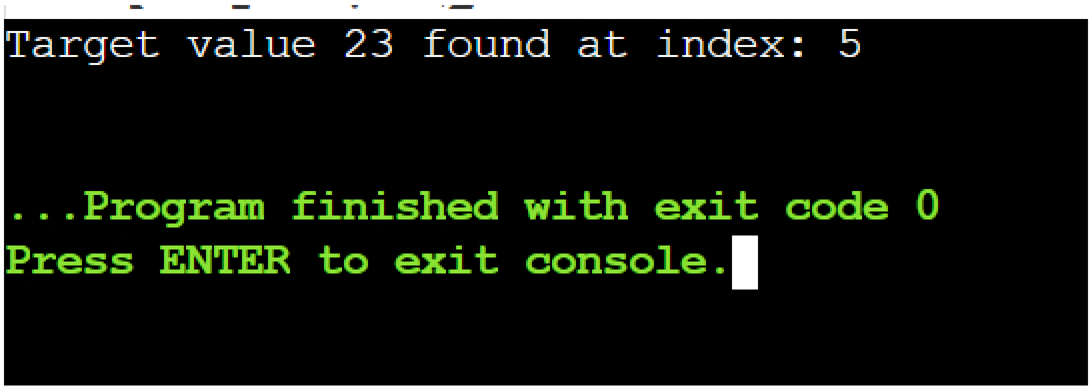
DAA ASSIGNMENT 1

1) BS

```
#include <iostream>
using namespace std;
int BS(int arr[], int size, int reqd) {
    int left = 0, right = size - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == reqd) {
            return mid;
        } else if (arr[mid] < reqd) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}
int main() {
    int arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91};
    int reqd = 23;
    int size = sizeof(arr) / sizeof(arr[0]);

    int result = BS(arr, size, reqd);

    if (result != -1) {
        cout << "Target value " << reqd << " found at index: " << result << endl;
    } else {
        cout << "Target not found" << endl;
    }
}
```



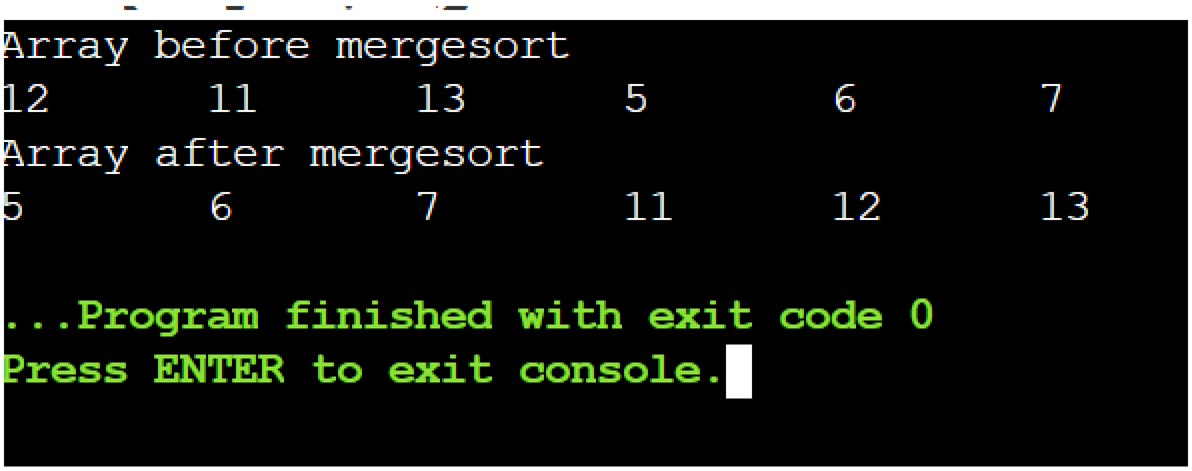
```
Target value 23 found at index: 5


...Program finished with exit code 0
Press ENTER to exit console.
```

2) MERGE SORT

```
#include <iostream>
```

```
using namespace std;
void merge(int arr[], int low, int mid, int high) {
    int n1 = mid - low + 1;
    int n2 = high - mid;
    int leftArr[n1], rightArr[n2];
    for (int i = 0; i < n1; i++) {
        leftArr[i] = arr[low + i];
    }
    for (int i = 0; i < n2; i++) {
        rightArr[i] = arr[mid + 1 + i];
    }
    int i = 0, j = 0, k = low;
    while (i < n1 && j < n2) {
        if (leftArr[i] <= rightArr[j]) {
            arr[k] = leftArr[i];
            i++;
        } else {
            arr[k] = rightArr[j];
            j++;
        }
        k++;
    }
    while (i < n1) {
        arr[k] = leftArr[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = rightArr[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int low, int high) {
    if (low < high) {
        int mid = low + (high - low) / 2;
        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);
        merge(arr, low, mid, high);
    }
}
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout<<"Array before mergesort \n";
    for (int i = 0; i < n; i++) {
        cout << arr[i]<<"\t";
    }
    cout<<endl;
    cout<<"Array after mergesort \n";
    mergeSort(arr, 0, n - 1);
    for (int i = 0; i < n; i++) {
        cout << arr[i]<<"\t";
    }
    return 0;
}
```

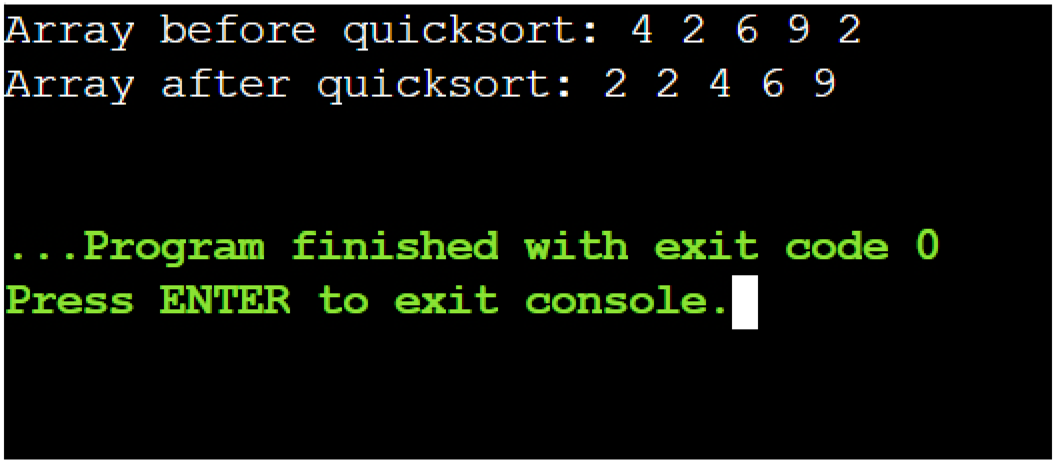


```
Array before mergesort
12      11      13      5       6       7
Array after mergesort
5       6       7       11      12      13

...Program finished with exit code 0
Press ENTER to exit console.
```

3) QUICKSORT

```
#include <iostream>
using namespace std;
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) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
int main() {
    int arr[] = {4, 2, 6, 9, 2};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "Array before quicksort: ";
    printArray(arr, n);
    quickSort(arr, 0, n - 1);
    cout << "Array after quicksort: ";
    printArray(arr, n);
    return 0;
}
```



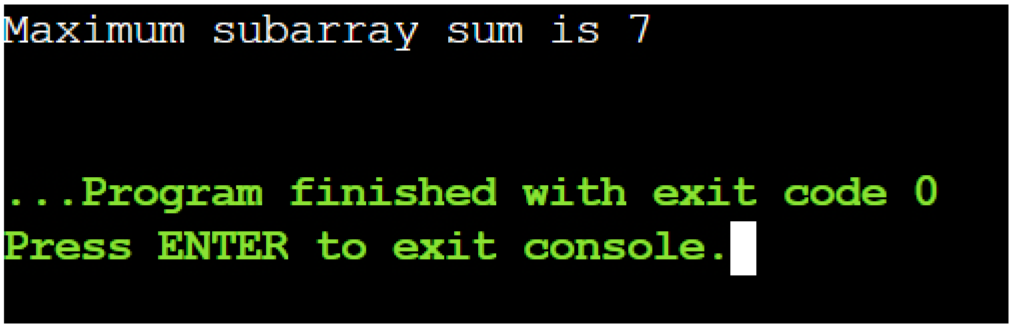
```
Array before quicksort: 4 2 6 9 2
Array after quicksort: 2 2 4 6 9


...Program finished with exit code 0
Press ENTER to exit console.
```

4)

MAXSUBARRAYSUM

```
#include <iostream>
using namespace std;
int maxSubArraySum(int arr[], int n) {
    int maxSum = arr[0], currentSum = arr[0];
    for (int i = 1; i < n; i++) {
        currentSum = max(arr[i], currentSum + arr[i]);
        maxSum = max(maxSum, currentSum);
    }
    return maxSum;
}
int main() {
    int arr[] = {-2, -5, 6, -2, -3, 1, 5, -6};
    int n = sizeof(arr) / sizeof(arr[0]);
    int result = maxSubArraySum(arr, n);
    cout << "Maximum subarray sum is " << result << endl;
    return 0;
}
```



```
Maximum subarray sum is 7


...Program finished with exit code 0
Press ENTER to exit console.
```