DAA – LAB 2

Name : Madhuramsinh Solanki

Reg no: 22BRS1327

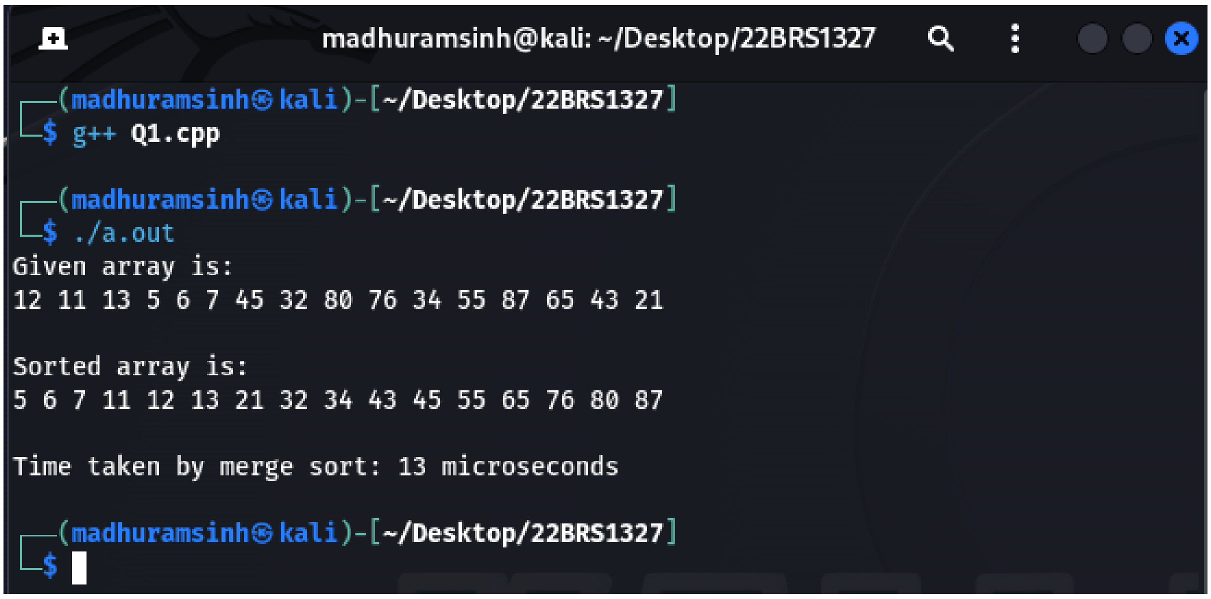
Q1) Merge Sort using STL LIbraries

Code:

```
#include <iostream>

#include <vector>

#include <chrono>

using namespace std;

using namespace std::chrono;

void merge(vector<int>& arr, int left, int mid, int right) {

   int size1 = mid - left + 1;

   int size2 = right - mid;

   vector<int> leftArray(size1), rightArray(size2);

   for (int i = 0; i < size1; i++)

      leftArray[i] = arr[left + i];

   for (int j = 0; j < size2; j++)

      rightArray[j] = arr[mid + 1 + j];

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

   while (i < size1 && j < size2) {

      if (leftArray[i] <= rightArray[j]) {

         arr[k] = leftArray[i];

         i++;

      } else {

         arr[k] = rightArray[j];

         j++;

      }

      k++;

   }

   while (i < size1) {

      arr[k] = leftArray[i];

      i++;
```

```
        k++;
    }
    while (j < size2) {
        arr[k] = rightArray[j];
        j++;
        k++;
    }
}
void mergeSort(vector<int>& arr, int left, int right) {
    if (left >= right) return;
    int mid = left + (right - left) / 2;
    mergeSort(arr, left, mid);
    mergeSort(arr, mid + 1, right);
    merge(arr, left, mid, right);
}
void printArray(const vector<int>& arr) {
    for (int num : arr) {
        cout << num << " ";
    }
    cout << endl;
}
int main() {
    vector<int> numbers = {12, 11, 13, 5, 6, 7, 45, 32, 80, 76, 34, 55, 87, 65, 43, 21};
    int size = numbers.size();
    cout << "Given array is: \n";
    printArray(numbers);
    auto startTime = high_resolution_clock::now();
    mergeSort(numbers, 0, size - 1);
    auto endTime = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(endTime - startTime);
    cout << "\nSorted array is: \n";
    printArray(numbers);
    cout << "\nTime taken by merge sort: " << duration.count() << " microseconds" << endl;
    return 0;
}
```

Output:



```
madhuramsinh@kali: ~/Desktop/22BRS1327

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ g++ Q1.cpp

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ ./a.out
Given array is:
12 11 13 5 6 7 45 32 80 76 34 55 87 65 43 21

Sorted array is:
5 6 7 11 12 13 21 32 34 43 45 55 65 76 80 87

Time taken by merge sort: 13 microseconds

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ 
```

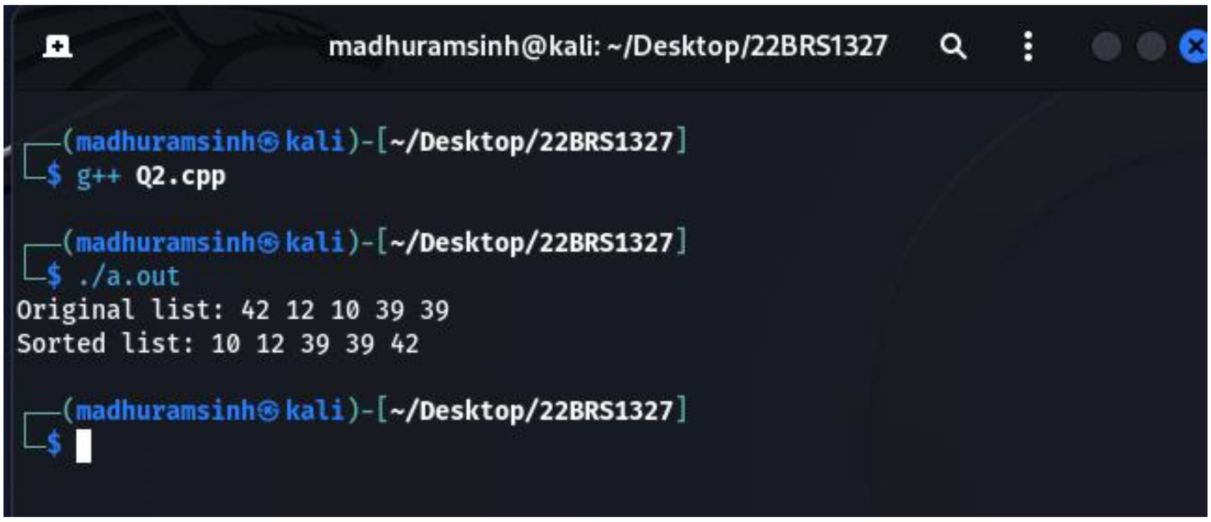
Q2) Merge Sort Using Linked List

Code:

```
#include <iostream>

using namespace std;

struct ListNode {

   int value;

   ListNode* next;

   ListNode(int x) : value(x), next(nullptr) {}

};

ListNode* merge(ListNode* list1, ListNode* list2) {

   if (list1 == nullptr) return list2;

   if (list2 == nullptr) return list1;

   ListNode* mergedHead = nullptr;

   if (list1->value <= list2->value) {

      mergedHead = list1;

      mergedHead->next = merge(list1->next, list2);

   } else {

      mergedHead = list2;

      mergedHead->next = merge(list1, list2->next);

   }

   return mergedHead;
}

ListNode* findMiddle(ListNode* head) {

    if (head == nullptr || head->next == nullptr) return head;

    ListNode* slowPointer = head;

    ListNode* fastPointer = head->next;

    while (fastPointer != nullptr && fastPointer->next != nullptr) {

        slowPointer = slowPointer->next;

        fastPointer = fastPointer->next->next;

    }

    ListNode* middle = slowPointer->next;

    slowPointer->next = nullptr;

    return middle;

}

ListNode* mergeSort(ListNode* head) {

    if (head == nullptr || head->next == nullptr) return head;

    ListNode* middle = findMiddle(head);

    ListNode* leftHalf = mergeSort(head);

    ListNode* rightHalf = mergeSort(middle);

    return merge(leftHalf, rightHalf);

}

void printList(ListNode* head) {

    ListNode* currentNode = head;

    while (currentNode != nullptr) {

        cout << currentNode->value << " ";

        currentNode = currentNode->next;

    }

    cout << endl;

}

void deleteList(ListNode* head) {

    while (head != nullptr) {

        ListNode* temp = head;

        head = head->next;

        delete temp;

    }

}
```

```
int main() {
    ListNode* head = new ListNode(42);
    head->next = new ListNode(12);
    head->next->next = new ListNode(10);
    head->next->next->next = new ListNode(39);
    head->next->next->next->next = new ListNode(39);
    cout << "Original list: ";
    printList(head);
    head = mergeSort(head);
    cout << "Sorted list: ";
    printList(head);
    deleteList(head);
    return 0;
}
```

Output:



```
┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ g++ Q2.cpp

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ ./a.out
Original list: 42 12 10 39 39
Sorted list: 10 12 39 39 42

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ 
```

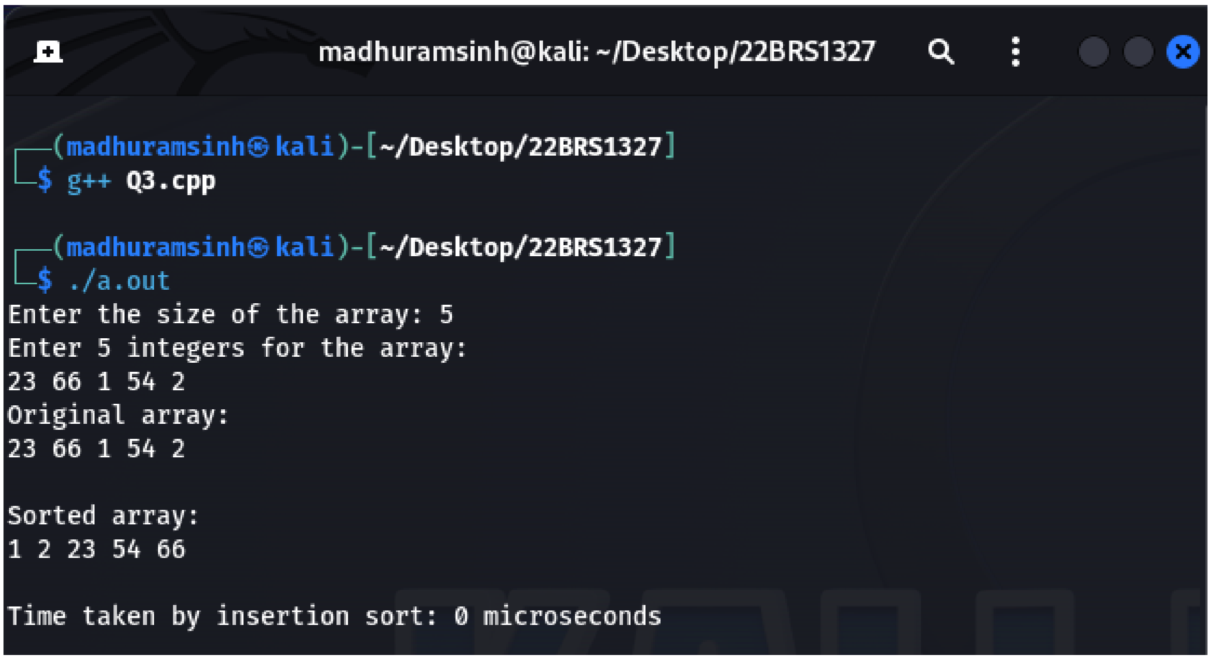
Q3) Insertion Sort with Time Computation

Code:

```
#include <iostream>

#include <vector>

#include <chrono>

#include <algorithm>

using namespace std;

using namespace std::chrono;

void insertionSort(vector<int>& numbers) {

   int size = numbers.size();

   for (int i = 1; i < size; ++i) {

      int key = numbers[i];

      int j = i - 1;

      while (j >= 0 && numbers[j] > key) {

         numbers[j + 1] = numbers[j];

         j--;

      }

      numbers[j + 1] = key;

   }

}

void printArray(const vector<int>& numbers) {

   for (int num : numbers) {

      cout << num << " ";

   }

   cout << endl;

}

int main() {

   int size;

   cout << "Enter the size of the array: ";

   cin >> size;

   vector<int> numbers(size);

   cout << "Enter " << size << " integers for the array:" << endl;

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

      cin >> numbers[i];
```

```
    }
    cout << "Original array:" << endl;
    printArray(numbers);
    auto startTime = high_resolution_clock::now();
    insertionSort(numbers);
    auto endTime = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(endTime - startTime);
    cout << "\nSorted array:" << endl;
    printArray(numbers);
    cout << "\nTime taken by insertion sort: " << duration.count() << " microseconds" << endl;
    return 0;
}
```

Output:



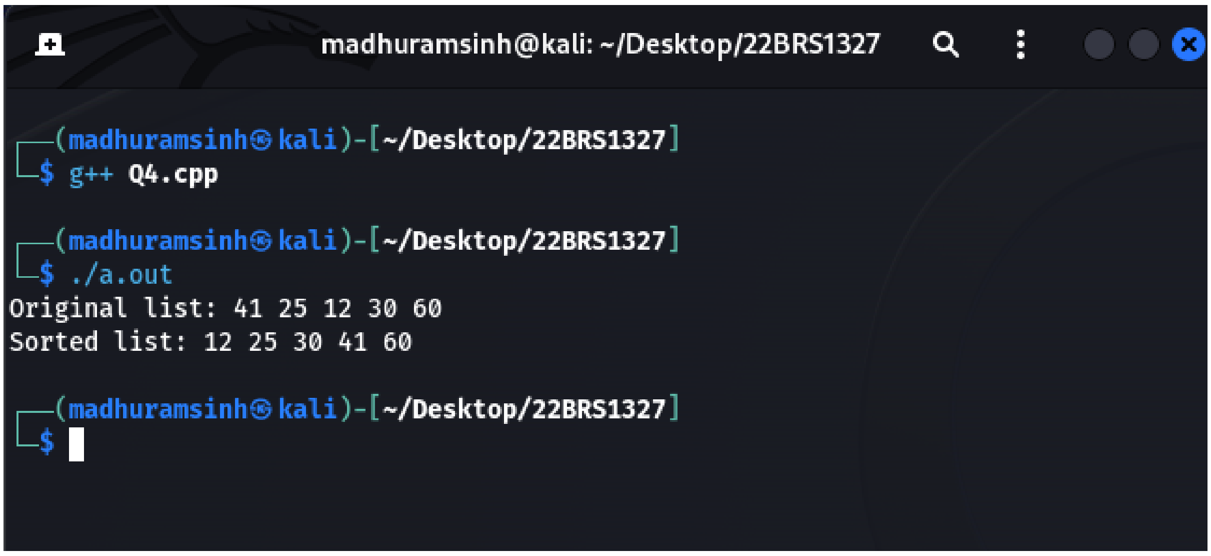
```
┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ g++ Q3.cpp

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ ./a.out
Enter the size of the array: 5
Enter 5 integers for the array:
23 66 1 54 2
Original array:
23 66 1 54 2

Sorted array:
1 2 23 54 66

Time taken by insertion sort: 0 microseconds
```

Q4) Insertion Sort Using Linked List

Code:

```
#include <iostream>

using namespace std;

struct ListNode {

   int value;

   ListNode* next;

   ListNode(int x) : value(x), next(nullptr) {}

};

ListNode* insertionSortList(ListNode* head) {

   if (head == nullptr || head->next == nullptr) return head;

   ListNode* dummy = new ListNode(0);

   ListNode* current = head;

   while (current != nullptr) {

      ListNode* prev = dummy;

      while (prev->next != nullptr && prev->next->value < current->value) {

         prev = prev->next;

      }

      ListNode* nextNode = current->next;

      current->next = prev->next;

      prev->next = current;

      current = nextNode;

   }

   return dummy->next;

}

void printList(ListNode* head) {

   ListNode* current = head;

   while (current != nullptr) {

      cout << current->value << " ";

      current = current->next;

   }

   cout << endl;

}

void deleteList(ListNode* head) {
   while (head != nullptr) {
      ListNode* temp = head;
      head = head->next;
      delete temp;
   }
}
int main() {
   ListNode* head = new ListNode(41);
   head->next = new ListNode(25);
   head->next->next = new ListNode(12);
   head->next->next->next = new ListNode(30);
   head->next->next->next->next = new ListNode(60);
   cout << "Original list: ";
   printList(head);
   head = insertionSortList(head);
   cout << "Sorted list: ";
   printList(head);
   deleteList(head);
   return 0;
}
```

Output:



```
┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ g++ Q4.cpp

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ ./a.out
Original list: 41 25 12 30 60
Sorted list: 12 25 30 41 60

┌──(madhuramsinh㉿kali)-[~/Desktop/22BRS1327]
└─$ 
```