Algorithm and Design

Name: Ahmed Kasteer

Roll Number: 20F-0336

Section: 4D

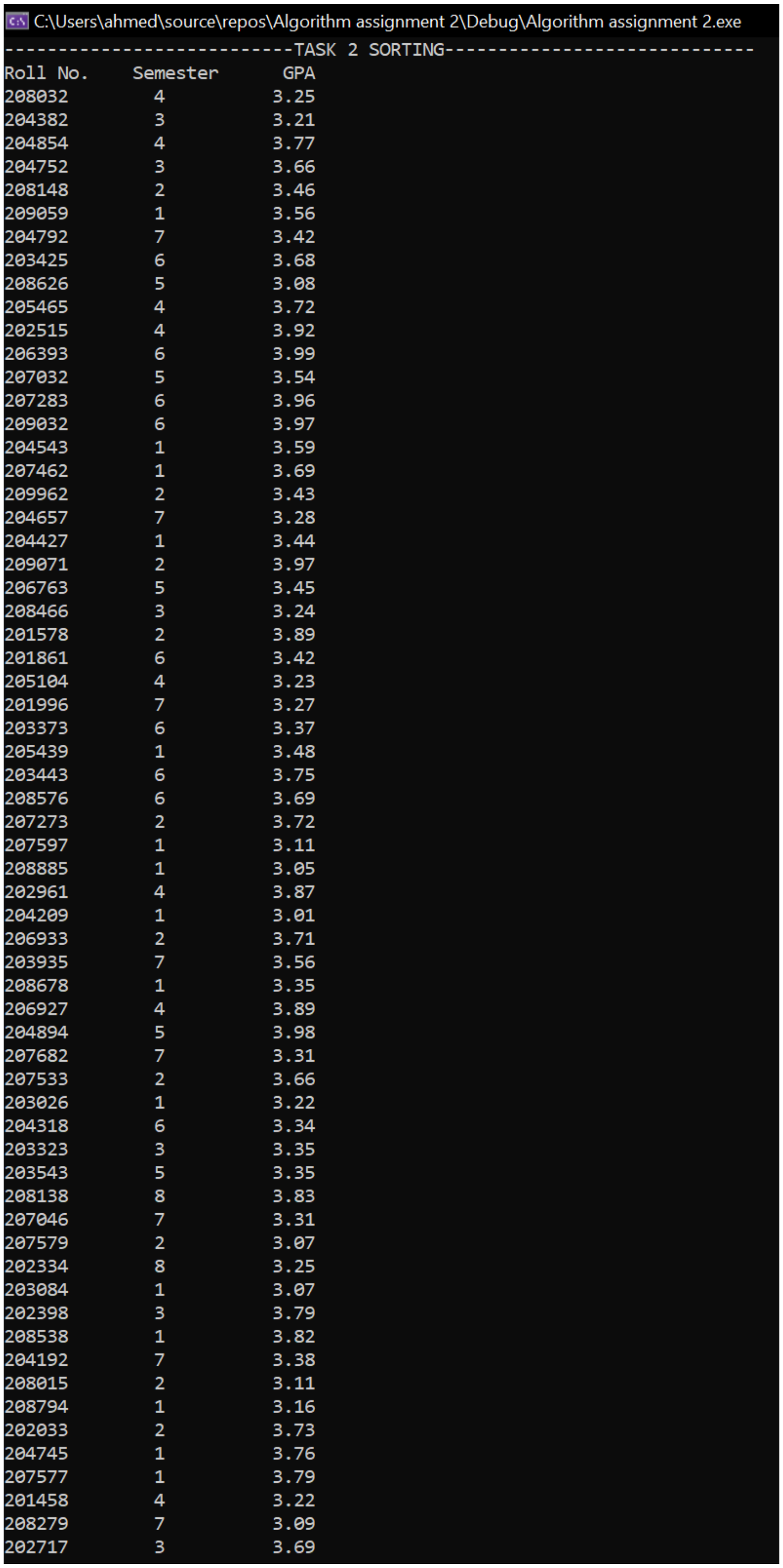
Task 2

Code written in C++

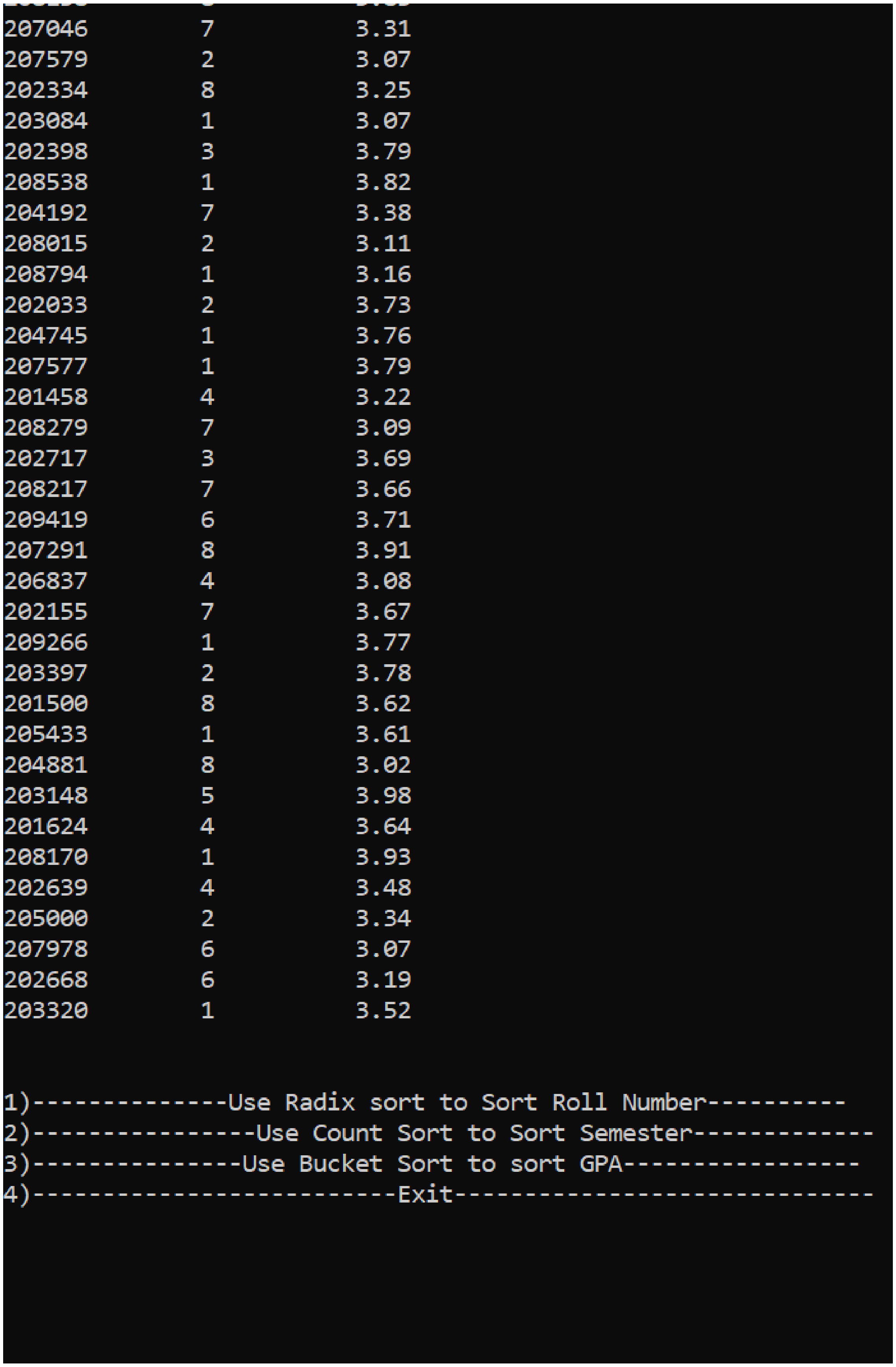
```
#include <iostream>
#include <fstream>
#include<vector>
#include<string>
#include<algorithm>
#define RANGE 255
using namespace std;

int getMax(int arr[], int n)
{
    int mx = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > mx)
            mx = arr[i];
    return mx;
}
void countSort_redix(int arr[], int n, int exp)
{
    int output[81];
    int i, count[10] = { 0 };
    for (i = 0; i < n; i++)
        count[(arr[i] / exp) % 10]++;
    for (i = 1; i < 10; i++)
        count[i] += count[i - 1];
    for (i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }
    for (i = 0; i < n; i++)
        arr[i] = output[i];
}
void radixsort(int arr[], int n)
{
    int m = getMax(arr, n);
    for (int exp = 1; m / exp > 0; exp *= 10)
        countSort_redix(arr, n, exp);
}


void countSort1(int input_array[], int s, int r)
{
    int output_array[81];
    int count_array[10] = { 0 };


    for (int i = 0; i < r; i++)
        count_array[i] = 0;


    for (int i = 0; i < s; i++)
        ++count_array[input_array[i]];


    for (int i = 1; i < r; i++)
        count_array[i] = count_array[i] + count_array[i - 1];


    for (int i = 0; i < s; i++)
        output_array[--count_array[input_array[i]]] = input_array[i];


    for (int i = 0; i < s; i++)
        input_array[i] = output_array[i];
    for (int i = 0; i < 81; i++)
    {
        cout << input_array[i] << endl;
    }
}


void bucketSort(float* array, int size) {
    vector<float> bucket[81];
    for (int i = 0; i < size; i++) {
        bucket[int(size * array[i])].push_back(array[i]);
    }
    for (int i = 0; i < size; i++) {
        sort(bucket[i].begin(), bucket[i].end());
    }
    int index = 0;
    for (int i = 0; i < size; i++) {
        while (!bucket[i].empty()) {
            array[index++] = *(bucket[i].begin());
            bucket[i].erase(bucket[i].begin());
        }
    }
```

```
}


int main() {

    int i = 0;
    ifstream File1("task2.txt");
    ofstream file("record.txt");

    int roll_no, roll[81];
    int semster, semester_name[81];
    float cgpa, gpa[81];
    cout << "---------------------------TASK 2 SORTING-----------------------------" <<
endl;
    cout << "Roll No." << "    Semester   " << "   GPA" << endl;
    while (File1 >> roll_no >> semster >> cgpa) {
        roll[i] = roll_no;
        semester_name[i] = semster;
        gpa[i] = cgpa;
        cout << roll_no << "         " << semster << "           " << cgpa << endl;
        i++;
    }
    int no = 0, no1 = 0, no2 = 0, no3 = 0;
    while (1)
    {
        cout << endl << endl;
        cout << "1)--------------Use Radix sort to Sort Roll Number----------"<< endl;
        cout << "2)----------------Use Count Sort to Sort Semester-------------" <<
endl;;
        cout << "3)---------------Use Bucket Sort to sort GPA-----------------"  << endl;
        cout << "4)-------------------------Exit------------------------------"<< endl;
        cin >> no;
        if (no == 1)
        {
            if (no1 != 1)
            {
                radixsort(roll, 81);
                for (int i = 0; i < 81; i++)
                {
                    file << roll[i] << "\n";
                    cout << roll[i] << "\n";

                }
                no1 = 1;
            }
            else
            {
                cout << "------------------Method used before and Sorted-----------------
-----" << endl;
            }
        }
        else if (no == 2)
        {
            if (no2 != 2)
            {
                countSort1(semester_name, 81, 10);
                for (int i = 0; i < 81; i++)
                {
                    file << semester_name[i] << "\n";
                }
                no2 = 2;
            }
            else
            {
                cout << "------------------Method used before and sorted.----------------
-----" << endl;
            }
        }
        else if (no == 3)
        {
            if (no3 != 3)
            {
                bucketSort(gpa, 81);
                for (int i = 0; i < 81; i++)
                {
                    file << gpa[i] << "\n";
                    cout << gpa[i] << endl;
                }
                no3 = 3;
            }
            else
                cout << "------------------Method used before and sorted.----------------
-----" << endl;
        }
        else
        {
            break;
        }
    }

}
```

Screenshots

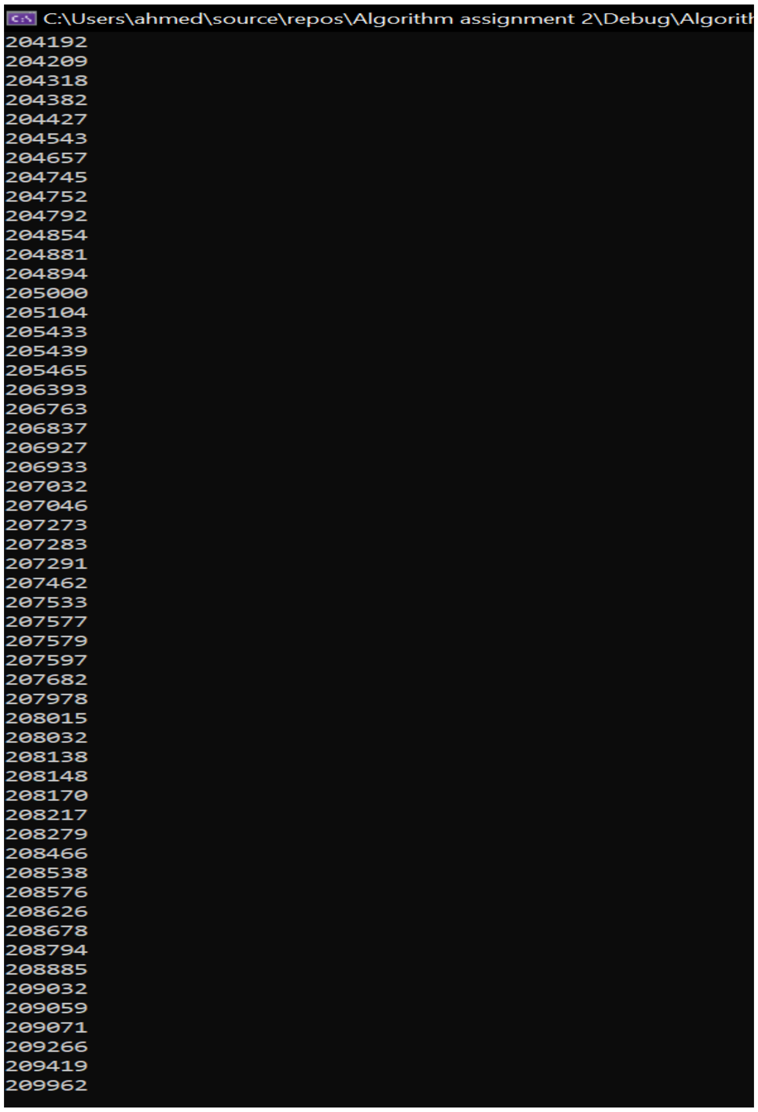

C:\Users\ahmed\source\repos\Algorithm assignment 2\Debug\Algorithm assignment 2.exe

```
-------------------------TASK 2 SORTING---------------------------
Roll No.    Semester      GPA
208032        4          3.25
204382        3          3.21
204854        4          3.77
204752        3          3.66
208148        2          3.46
209059        1          3.56
204792        7          3.42
203425        6          3.68
208626        5          3.08
205465        4          3.72
202515        4          3.92
206393        6          3.99
207032        5          3.54
207283        6          3.96
209032        6          3.97
204543        1          3.59
207462        1          3.69
209962        2          3.43
204657        7          3.28
204427        1          3.44
209071        2          3.97
206763        5          3.45
208466        3          3.24
201578        2          3.89
201861        6          3.42
205104        4          3.23
201996        7          3.27
203373        6          3.37
205439        1          3.48
203443        6          3.75
208576        6          3.69
207273        2          3.72
207597        1          3.11
208885        1          3.05
202961        4          3.87
204209        1          3.01
206933        2          3.71
203935        7          3.56
208678        1          3.35
206927        4          3.89
204894        5          3.98
207682        7          3.31
207533        2          3.66
203026        1          3.22
204318        6          3.34
203323        3          3.35
203543        5          3.35
208138        8          3.83
207046        7          3.31
207579        2          3.07
202334        8          3.25
203084        1          3.07
202398        3          3.79
208538        1          3.82
204192        7          3.38
208015        2          3.11
208794        1          3.16
202033        2          3.73
204745        1          3.76
207577        1          3.79
201458        4          3.22
208279        7          3.09
202717        3          3.69
```



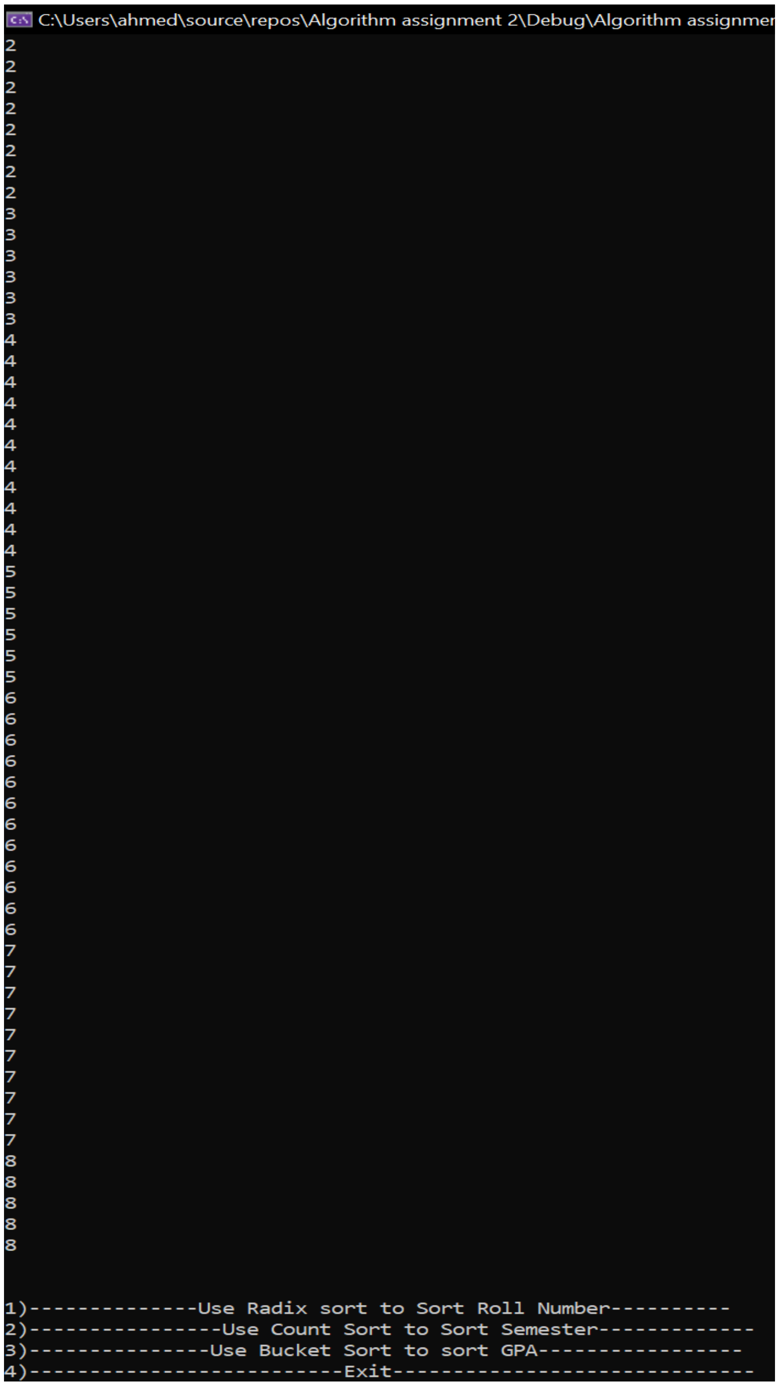
```
207046        7          3.31
207579        2          3.07
202334        8          3.25
203084        1          3.07
202398        3          3.79
208538        1          3.82
204192        7          3.38
208015        2          3.11
208794        1          3.16
202033        2          3.73
204745        1          3.76
207577        1          3.79
201458        4          3.22
208279        7          3.09
202717        3          3.69
208217        7          3.66
209419        6          3.71
207291        8          3.91
206837        4          3.08
202155        7          3.67
209266        1          3.77
203397        2          3.78
201500        8          3.62
205433        1          3.61
204881        8          3.02
203148        5          3.98
201624        4          3.64
208170        1          3.93
202639        4          3.48
205000        2          3.34
207978        6          3.07
202668        6          3.19
203320        1          3.52


1)--------------Use Radix sort to Sort Roll Number----------
2)----------------Use Count Sort to Sort Semester-------------
3)---------------Use Bucket Sort to sort GPA-----------------
4)-------------------------Exit------------------------------
```

Here Roll No. Sorted using Radix Sort:



```
C:\Users\ahmed\source\repos\Algorithm assignment 2\Debug\Algorith
204192
204209
204318
204382
204427
204543
204657
204745
204752
204792
204854
204881
204894
205000
205104
205433
205439
205465
206393
206763
206837
206927
206933
207032
207046
207273
207283
207291
207462
207533
207577
207579
207597
207682
207978
208015
208032
208138
208148
208170
208217
208279
208466
208538
208576
208626
208678
208794
208885
209032
209059
209071
209266
209419
209962
```

Count Sort used for Sorting Semester:


C:\Users\ahmed\source\repos\Algorithm assignment 2\Debug\Algorithm assignmer

```
2
2
2
2
2
2
2
2
3
3
3
3
3
3
4
4
4
4
4
4
4
4
4
4
4
5
5
5
5
5
5
6
6
6
6
6
6
6
6
6
6
6
6
7
7
7
7
7
7
7
7
7
7
8
8
8
8
8


1)--------------Use Radix sort to Sort Roll Number----------
2)----------------Use Count Sort to Sort Semester-------------
3)---------------Use Bucket Sort to sort GPA-----------------
4)--------------------------Exit------------------------------
```

TASK 3

Code written in C++

```
#include <iostream>
```

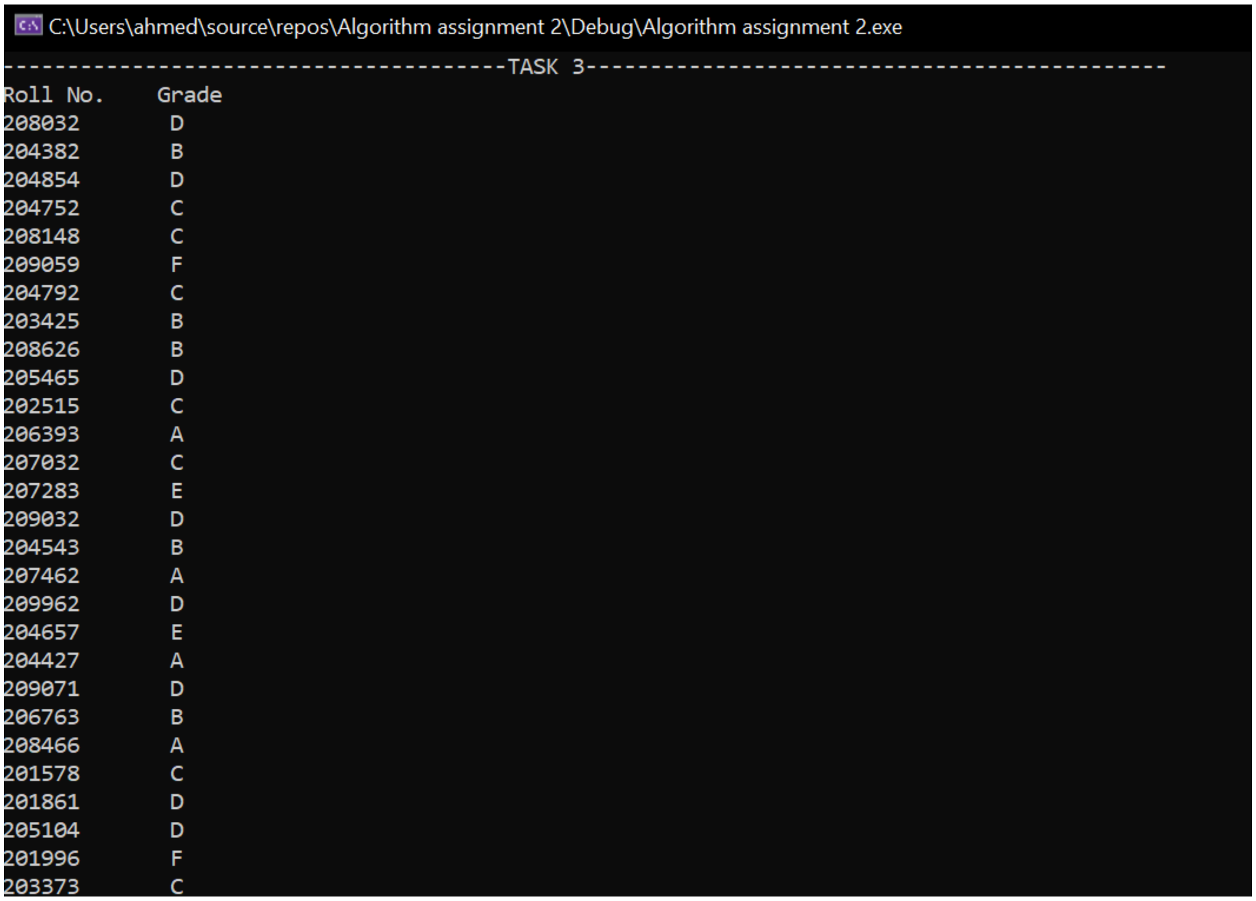
```cpp
#include <fstream>
#include<vector>
#include<string>
#include<algorithm>
#define RANGE 255
using namespace std;

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) {
        if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {
            array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
            indexOfSubArrayOne++;
        }
        else {
            array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
            indexOfSubArrayTwo++;
        }
        indexOfMergedArray++;
    }

    while (indexOfSubArrayOne < subArrayOne) {
        array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
        indexOfSubArrayOne++;
        indexOfMergedArray++;
    }

    while (indexOfSubArrayTwo < subArrayTwo) {
        array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
        indexOfSubArrayTwo++;
        indexOfMergedArray++;
    }
}
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 me(char array[], char const left, char const mid, char 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) {
            if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {
                array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
                indexOfSubArrayOne++;
            }
            else {
                array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                indexOfSubArrayTwo++;
            }
            indexOfMergedArray++;
        }

        while (indexOfSubArrayOne < subArrayOne) {
            array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
            indexOfSubArrayOne++;
            indexOfMergedArray++;
        }

        while (indexOfSubArrayTwo < subArrayTwo) {
            array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
            indexOfSubArrayTwo++;
            indexOfMergedArray++;
        }
    }


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

        auto mid = begin + (end - begin) / 2;
        mergeSort_grade(array, begin, mid);
        mergeSort_grade(array, mid + 1, end);
        me(array, begin, mid, end);
```

```
}
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++;
    }


    int pivotIndex = start + count;
    swap(arr[pivotIndex], arr[start]);


    int i = start, j = end;

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

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

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

        if (i < pivotIndex && j > pivotIndex) {
            swap(arr[i++], arr[j--]);
        }
    }

    return pivotIndex;
}

void quickSort(int arr[], int start, int end)
{


    if (start >= end)
        return;


    int p = partition(arr, start, end);


    quickSort(arr, start, p - 1);


    quickSort(arr, p + 1, end);
}
int part(char arr[], char start, char end)
{

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


    int pivotIndex = start + count;
    swap(arr[pivotIndex], arr[start]);


    int i = start, j = end;

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

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

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

        if (i < pivotIndex && j > pivotIndex) {
            swap(arr[i++], arr[j--]);
        }
    }

    return pivotIndex;
}

void quickSort_grade(char arr[], int start, int end)
{


    if (start >= end)
        return;


    int p = part(arr, start, end);


    quickSort_grade(arr, start, p - 1);


    quickSort_grade(arr, p + 1, end);
}




int main() {

    int i = 0;
    ifstream file("task3.txt");
    int rol, rollno[81];
    char gra, grade[81];
    cout << "---------------------------------------TASK 3------------------------------
--------------" << endl;
    cout << "Roll No." << "    Grade" << endl;
    while (file >> rol >> gra) {
        rollno[i] = rol;

        grade[i] = gra;
        cout << rol << "        " << gra << endl;
        i++;
    }
    int input;
    while (1)
    {
        cout << " 1)--------------Merge Sort for Grade Sorting-------------------------"
<< endl;
        cout << " 2)--------------Quick Sort for Roll Number Sorting-------------------"
<< endl;
        cout << " 3)--------------Quick Sort for Grade Sorting-------------------------"
<< endl;
        cout << " 4)--------------Merge Sort for RollNumber Sorting-------------------"
<< endl;
        cin >> input;

        if (input == 1)
        {
            cout << "----------------Merge Sort for Grade--------------------" << endl;
            mergeSort_grade(grade, 0, 80);
            for (int i = 0; i < 81; i++)
            {
                cout << grade[i] << "\n";

            }
        }
        else if (input == 2)
        {
            cout << "----------------Quick Sort for Roll Number----------------" << endl;
            quickSort(rollno, 0, 80);
            for (int i = 0; i < 81; i++)
            {
                cout << rollno[i] << "\n";

            }
        }
        else if (input == 3)
        {
            cout << "----------------Quick Sort for Grade----------------" << endl;
            quickSort_grade(grade, 0, 80);
            for (int i = 0; i < 81; i++)
            {
                cout << grade[i] << "\n";

            }
        }
        else if (input == 4)
        {
            cout << "----------------Merge Sort for Roll Number----------------" << endl;
            mergeSort(rollno, 0, 80);
            for (int i = 0; i < 81; i++)
```

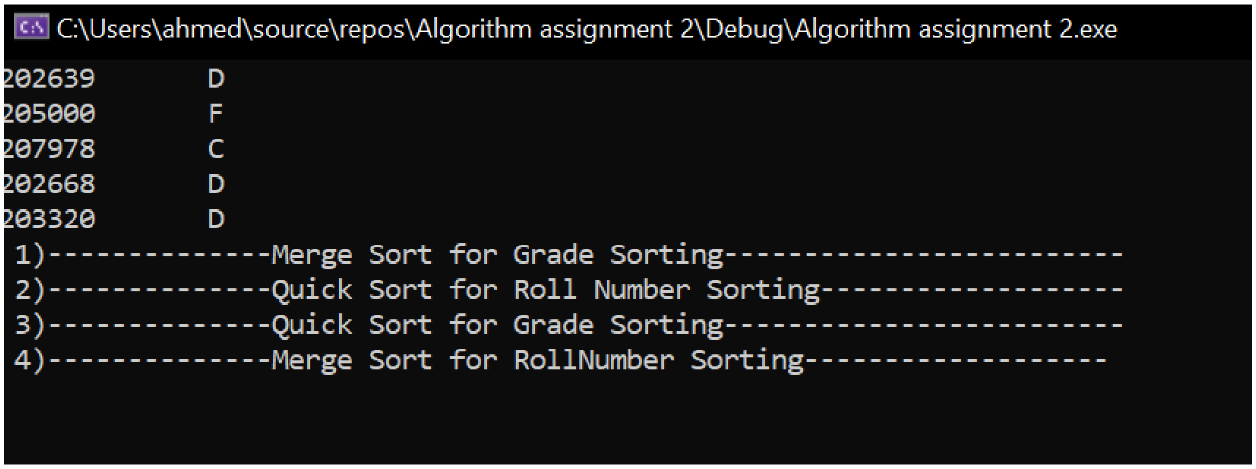
```
                {
                    cout << rollno[i] << "\n";

                }
            }
            else
            {
                break;
            }
        }
                                                    }
```

Screenshots

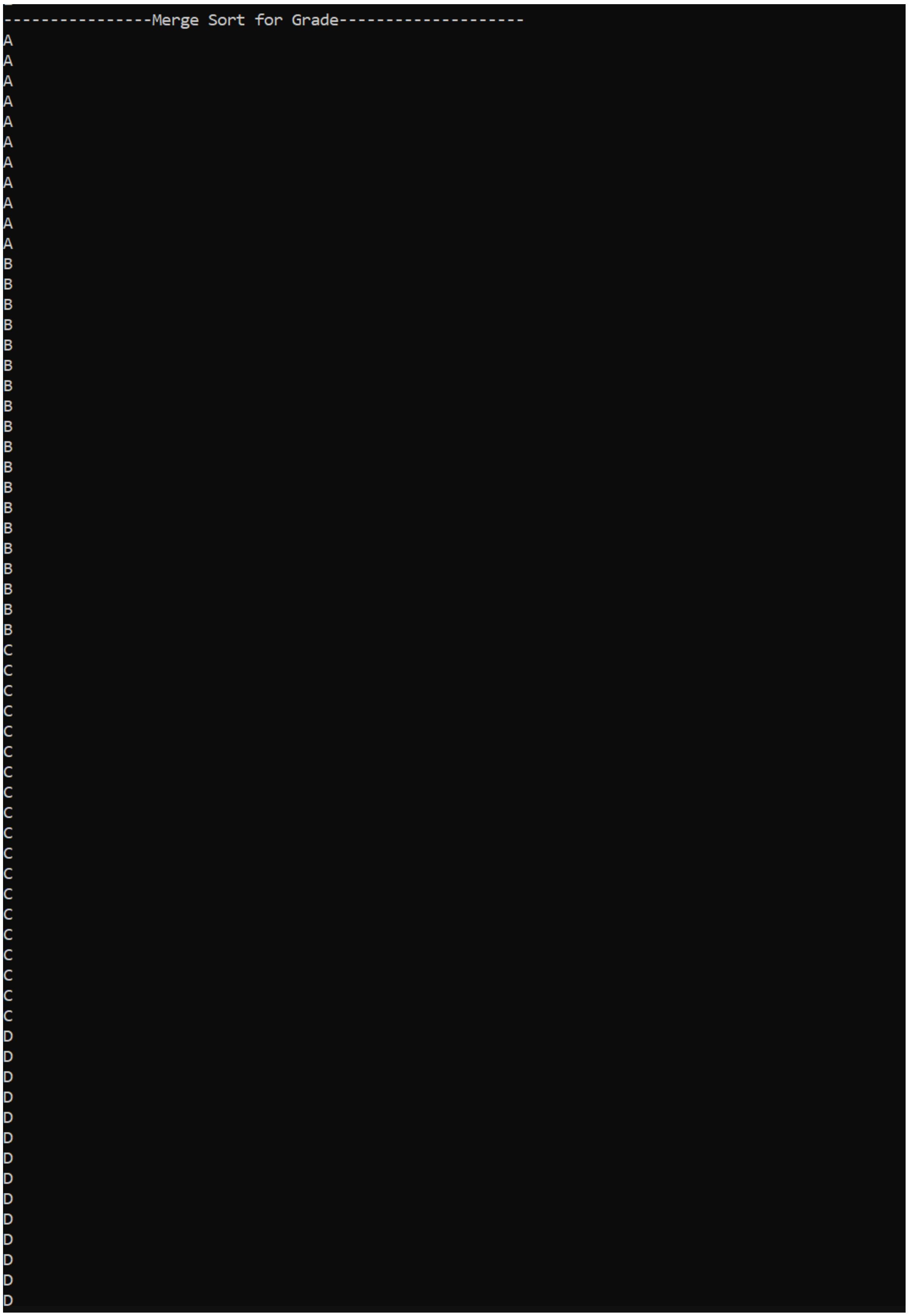


C:\Users\ahmed\source\repos\Algorithm assignment 2\Debug\Algorithm assignment 2.exe

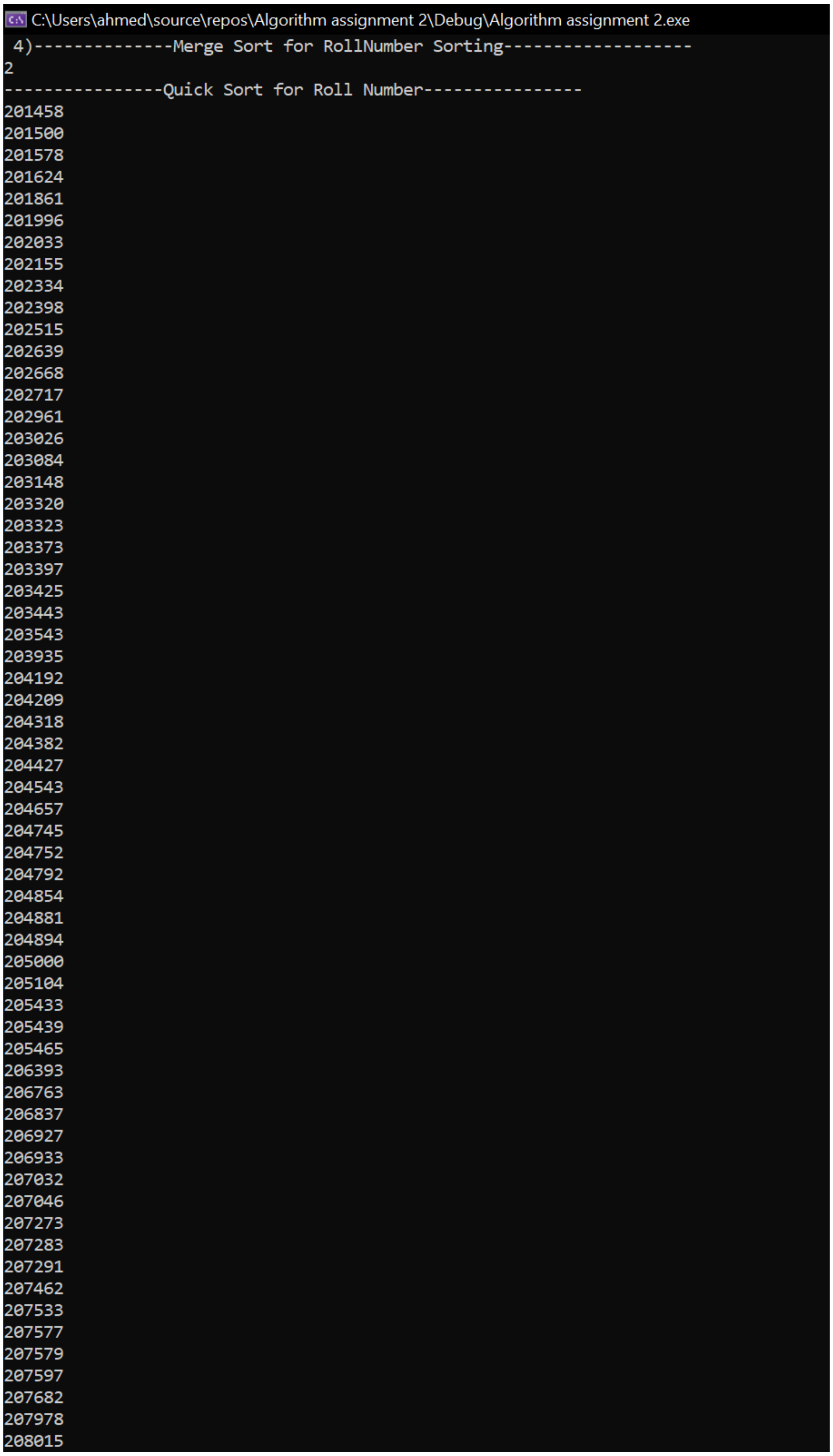
```
-------------------------------------TASK 3-----------------------------------------
Roll No.    Grade
208032       D
204382       B
204854       D
204752       C
208148       C
209059       F
204792       C
203425       B
208626       B
205465       D
202515       C
206393       A
207032       C
207283       E
209032       D
204543       B
207462       A
209962       D
204657       E
204427       A
209071       D
206763       B
208466       A
201578       C
201861       D
205104       D
201996       F
203373       C
```


C:\Users\ahmed\source\repos\Algorithm assignment 2\Debug\Algorithm assignment 2.exe

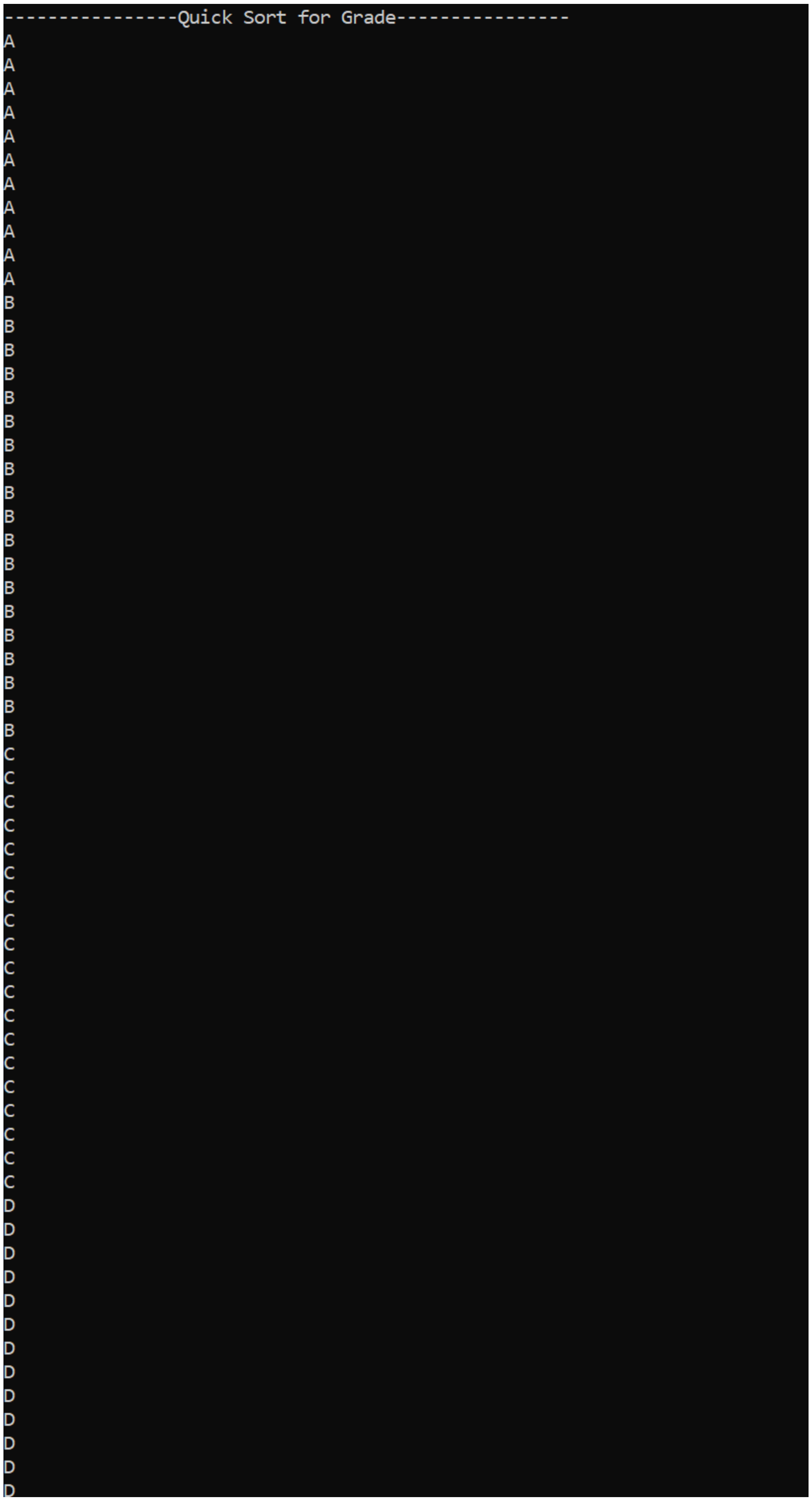
```
202639       D
205000       F
207978       C
202668       D
203320       D
 1)-------------Merge Sort for Grade Sorting------------------------
 2)-------------Quick Sort for Roll Number Sorting------------------
 3)-------------Quick Sort for Grade Sorting------------------------
 4)-------------Merge Sort for RollNumber Sorting------------------
```



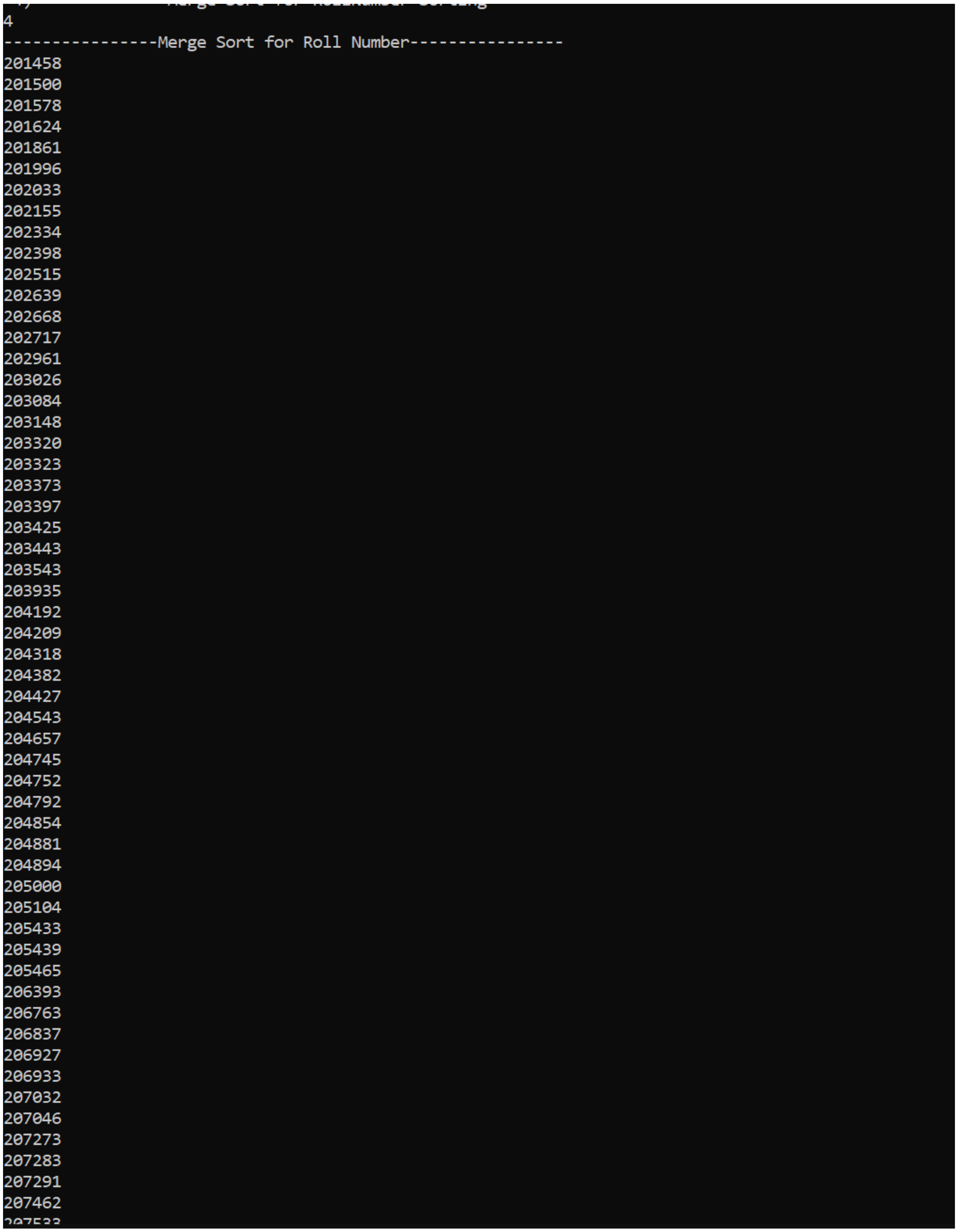
```
----------------Merge Sort for Grade--------------------
A
A
A
A
A
A
A
A
A
A
A
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
D
D
D
D
D
D
D
D
D
D
D
D
D
D
```



```
C:\Users\ahmed\source\repos\Algorithm assignment 2\Debug\Algorithm assignment 2.exe
 4)--------------Merge Sort for RollNumber Sorting-------------------
2
----------------Quick Sort for Roll Number----------------
201458
201500
201578
201624
201861
201996
202033
202155
202334
202398
202515
202639
202668
202717
202961
203026
203084
203148
203320
203323
203373
203397
203425
203443
203543
203935
204192
204209
204318
204382
204427
204543
204657
204745
204752
204792
204854
204881
204894
205000
205104
205433
205439
205465
206393
206763
206837
206927
206933
207032
207046
207273
207283
207291
207462
207533
207577
207579
207597
207682
207978
208015
```



```
----------------Quick Sort for Grade----------------
A
A
A
A
A
A
A
A
A
A
A
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
D
D
D
D
D
D
D
D
D
D
D
D
D
```



```
4)              Merge Sort for RollNumber Sorting
4
----------------Merge Sort for Roll Number----------------
201458
201500
201578
201624
201861
201996
202033
202155
202334
202398
202515
202639
202668
202717
202961
203026
203084
203148
203320
203323
203373
203397
203425
203443
203543
203935
204192
204209
204318
204382
204427
204543
204657
204745
204752
204792
204854
204881
204894
205000
205104
205433
205439
205465
206393
206763
206837
206927
206933
207032
207046
207273
207283
207291
207462
207533
```