DSA LAB 07

24K-0852 BSCS-3K Hasan Mujtaba

Task 1:

```cpp
#include <iostream>
#include <string>
using namespace std;

class DispatchRecord {
public:
    int DispatchID;
    string CustomerName;
    int DeliveryTime;

    void input() {
        cout << "Enter Dispatch ID: ";
        cin >> DispatchID;
        cin.ignore();
        cout << "Enter Customer Name: ";
        getline(cin, CustomerName);
        cout << "Enter Delivery Time (in minutes): ";
        cin >> DeliveryTime;
    }

    void display() const {
        cout << DispatchID << "\t" << CustomerName << "\t" << DeliveryTime <<
endl;
    }
};

int partition(DispatchRecord arr[], int low, int high) {
    int pivot = arr[high].DeliveryTime;
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (arr[j].DeliveryTime < pivot) {
            i++;
            swap(arr[i], arr[j]);
        }
    }
    swap(arr[i + 1], arr[high]);
```

```
    return i + 1;
}

void quickSort(DispatchRecord arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

int main() {
    int n = 5;
    DispatchRecord records[] = {
        {101, "Ali", 35},
        {102, "Sara", 20},
        {103, "Ahmed", 50},
        {104, "Noor", 15},
        {105, "Hasan", 25}
    };

    cout << "\n--- Records Before Sorting ---\n";
    cout << "ID\tName\tDeliveryTime\n";
    for (int i = 0; i < n; i++)
        records[i].display();

    quickSort(records, 0, n - 1);

    cout << "\n--- Records After Sorting (by Delivery Time) ---\n";
    cout << "ID\tName\tDeliveryTime\n";
    for (int i = 0; i < n; i++)
        records[i].display();

    return 0;
}
```

```
--- Records Before Sorting ---
ID      Name    DeliveryTime
101     Ali     35
102     Sara    20
103     Ahmed   50
104     Noor    15
105     Hasan   25

--- Records After Sorting (by Delivery Time) ---
ID      Name    DeliveryTime
104     Noor    15
102     Sara    20
105     Hasan   25
101     Ali     35
103     Ahmed   50
```

Task 2:

```cpp
#include <iostream>
using namespace std;

int getMax(int arr[], int n) {
    int maxVal = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > maxVal)
            maxVal = arr[i];
    }
    return maxVal;
}

void countingSort(int arr[], int n, int exp) {
    int output[100];
    int count[10] = {0};

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

```
        count[(arr[i] / exp) % 10]++;

    for (int i = 1; i < 10; i++)
        count[i] += count[i - 1];

    for (int i = n - 1; i >= 0; i--) {
        int digit = (arr[i] / exp) % 10;
        output[count[digit] - 1] = arr[i];
        count[digit]--;
    }

    for (int i = 0; i < n; i++)
        arr[i] = output[i];
}

void radixSort(int arr[], int n) {
    int maxVal = getMax(arr, n);

    for (int exp = 1; maxVal / exp > 0; exp *= 10) {
        cout << "\nSorting by digit place: " << exp << endl;
        countingSort(arr, n, exp);

        cout << "Intermediate result: ";
        for (int i = 0; i < n; i++)
            cout << arr[i] << " ";
        cout << endl;
    }
}

int main() {
    int n;
    cout << "Enter number of account numbers: ";
    cin >> n;

    int accounts[100];
    cout << "\nEnter " << n << " account numbers (7-digit):\n";
    for (int i = 0; i < n; i++)
        cin >> accounts[i];

    cout << "\n--- Account Numbers Before Sorting ---\n";
    for (int i = 0; i < n; i++)
        cout << accounts[i] << " ";
    cout << endl;

    radixSort(accounts, n);
```

```
    cout << "\n--- Account Numbers After Sorting ---\n";
    for (int i = 0; i < n; i++)
        cout << accounts[i] << " ";
    cout << endl;

    return 0;
}
```

```
Enter number of account numbers: 3

Enter 3 account numbers (7-digit):
1234567
7654321
86753409

--- Account Numbers Before Sorting ---
1234567 7654321 86753409

Sorting by digit place: 1
Intermediate result: 7654321 1234567 86753409

Sorting by digit place: 10
Intermediate result: 86753409 7654321 1234567

Sorting by digit place: 100
Intermediate result: 7654321 86753409 1234567

Sorting by digit place: 1000
Intermediate result: 86753409 7654321 1234567

Sorting by digit place: 10000
Intermediate result: 1234567 86753409 7654321

Sorting by digit place: 100000
Intermediate result: 1234567 7654321 86753409

Sorting by digit place: 1000000
Intermediate result: 1234567 86753409 7654321

Sorting by digit place: 10000000
Intermediate result: 1234567 7654321 86753409

--- Account Numbers After Sorting ---
1234567 7654321 86753409
```