Taskin Ahmad Al Faruqe

ID: 242910701126

Home Work-03

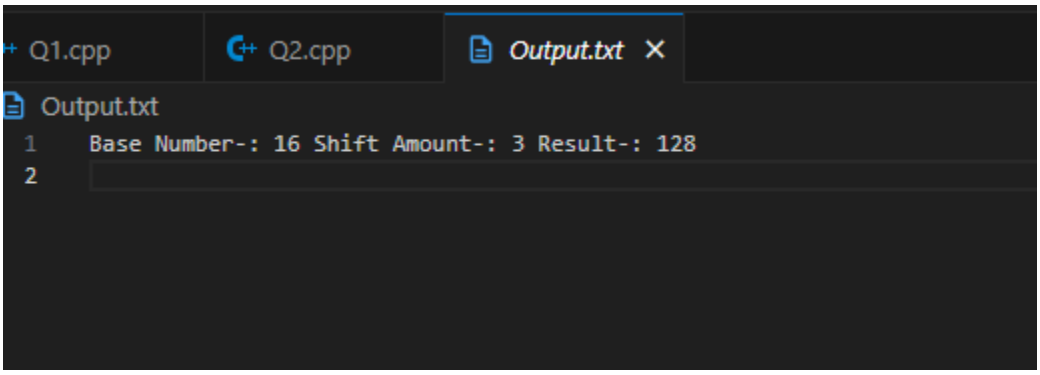
Question-01(Rotate Left K bits)

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

int rotateLeft(int n, int k) {
    int bits = sizeof(n) * 8;
    k = k % bits;
    return (n << k) | (n >> (bits - k));
}

int main() {
    int n = 16;
    int k = 2;
    int result = rotateLeft(n, k);
    cout << "Result: " << result << endl;
    return 0;
}
```

Output:



Question-02(Marge to List)

```cpp
#include <iostream>

using namespace std;

#define MAXSIZE 100

// Sequential List structure
struct SeqList {
    int data[MAXSIZE];
    int length;
};

// Initialize a SeqList
void initSeqList(SeqList &L) {
    L.length = 0;
}
```

```cpp
// Insert element
void insertElem(SeqList &L, int elem) {
    if (L.length >= MAXSIZE) {
        cout << "List is full!" << endl;
        return;
    }
    L.data[L.length++] = elem;
}

// Display list
void displaySeqList(const SeqList &L) {
    for (int i = 0; i < L.length; i++) {
        cout << L.data[i] << " ";
    }
    cout << endl;
}

// Bubble sort (Ascending order)
void sortSeqListAscending(SeqList &L) {
    for (int i = 0; i < L.length - 1; i++) {
        for (int j = 0; j < L.length - i - 1; j++) {
            if (L.data[j] > L.data[j + 1]) {  // Ascending
                int temp = L.data[j];
                L.data[j] = L.data[j + 1];
                L.data[j + 1] = temp;
            }
        }
    }
}
// Bubble sort (Descending order)
void sortSeqListDescending(SeqList &L) {
    for (int i = 0; i < L.length - 1; i++) {
        for (int j = 0; j < L.length - i - 1; j++) {
            if (L.data[j] < L.data[j + 1]) {  // Descending
                int temp = L.data[j];
                L.data[j] = L.data[j + 1];
                L.data[j + 1] = temp;
            }
        }
    }
}

// Merge two lists by combining then sorting ascending
SeqList mergeSeqListsAscending(const SeqList &A, const SeqList &B) {
    SeqList C;
    initSeqList(C);

    // Copy all elements
    for (int i = 0; i < A.length; i++)
        C.data[C.length++] = A.data[i];
    for (int j = 0; j < B.length; j++)
        C.data[C.length++] = B.data[j];

    // Sort ascending
    sortSeqListAscending(C);
    return C;
}
```

```cpp
// Merge two lists by combining then sorting descending
SeqList mergeSeqListsDescending(const SeqList &A, const SeqList &B) {
    SeqList C;
    initSeqList(C);

    // Copy all elements
    for (int i = 0; i < A.length; i++)
        C.data[C.length++] = A.data[i];
    for (int j = 0; j < B.length; j++)
        C.data[C.length++] = B.data[j];

    // Sort descending
    sortSeqListDescending(C);
    return C;
}
// Main function
int main() {
    SeqList list1, list2, mergedAsc, mergedDesc;

    initSeqList(list1);
    initSeqList(list2);

    // Example input
    insertElem(list1, 1);
    insertElem(list1, 5);
    insertElem(list1, 7);

    insertElem(list2, 2);
    insertElem(list2, 4);
    insertElem(list2, 6);
    insertElem(list2, 8);

    cout << "List 1: ";
    displaySeqList(list1);
    cout << "List 2: ";
    displaySeqList(list2);

    // Merge ascending
    mergedAsc = mergeSeqListsAscending(list1, list2);
    cout << "\nMerged List (Ascending): ";
    displaySeqList(mergedAsc);

    // Merge descending
    mergedDesc = mergeSeqListsDescending(list1, list2);
    cout << "Merged List (Descending): ";
    displaySeqList(mergedDesc);
    return 0;
}
```

Output: