SOURCE CODE

#include <stdio.h>

#include <string.h>

#define MAX\_COURSES 100

typedef struct {

int ID;

char cour\_id[20];

char course\_ref\_code[20];

char book\_title[100];

char book\_author[100];

char book\_details[200];

} CourseReference;

int linearSearch(CourseReference courses[], int size, int targetID);

int binarySearch(CourseReference courses[], int size, int targetID);

void bubbleSort(CourseReference courses[], int size);

void selectionSort(CourseReference courses[], int size);

void printCourses(CourseReference courses[], int size);

int main() {

CourseReference courses[MAX\_COURSES] = {

{1, "CSE101", "REF001", "Introduction to C", "Author A", "Details about C programming."},

{3, "CSE103", "REF003", "Algorithms", "Author C", "Details about algorithms."},

{2, "CSE102", "REF002", "Data Structures", "Author B", "Details about data structures."},

{5, "CSE105", "REF005", "Database Systems", "Author E", "Details about databases."},

{4, "CSE104", "REF004", "Operating Systems", "Author D", "Details about operating systems."}

};

int size = 5; // Number of courses

int targetID = 3; // ID to search for

int linearIndex = linearSearch(courses, size, targetID);

if (linearIndex != -1) {

printf("Linear Search: Course found at index %d\n", linearIndex);

} else {

printf("Linear Search: Course not found.\n");

}

bubbleSort(courses, size);

printf("\nCourses after Bubble Sort:\n");

printCourses(courses, size);

int binaryIndex = binarySearch(courses, size, targetID);

if (binaryIndex != -1) {

printf("Binary Search: Course found at index %d\n", binaryIndex);

} else {

printf("Binary Search: Course not found.\n");

}

selectionSort(courses, size);

printf("\nCourses after Selection Sort:\n");

printCourses(courses, size);

return 0;

}

int linearSearch(CourseReference courses[], int size, int targetID) {

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

if (courses[i].ID == targetID) {

return i; // Target found at index i

}

}

return -1; // Target not found

}

int binarySearch(CourseReference courses[], int size, int targetID) {

int low = 0;

int high = size - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (courses[mid].ID == targetID) {

return mid; // Target found at index mid

} else if (courses[mid].ID < targetID) {

low = mid + 1; // Search in the right half

} else {

high = mid - 1; // Search in the left half

}

}

return -1; // Target not found

}

void bubbleSort(CourseReference courses[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (courses[j].ID > courses[j + 1].ID) {

// Swap courses[j] and courses[j + 1]

CourseReference temp = courses[j];

courses[j] = courses[j + 1];

courses[j + 1] = temp;

}

}

}

}

void selectionSort(CourseReference courses[], int size) {

for (int i = 0; i < size - 1; i++) {

int minIndex = i;

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

if (courses[j].ID < courses[minIndex].ID) {

minIndex = j;

}

}

CourseReference temp = courses[minIndex];

courses[minIndex] = courses[i];

courses[i] = temp;

}

}

void printCourses(CourseReference courses[], int size) {

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

printf("ID: %d, Course ID: %s, Reference Code: %s, Title: %s, Author: %s, Details: %s\n",

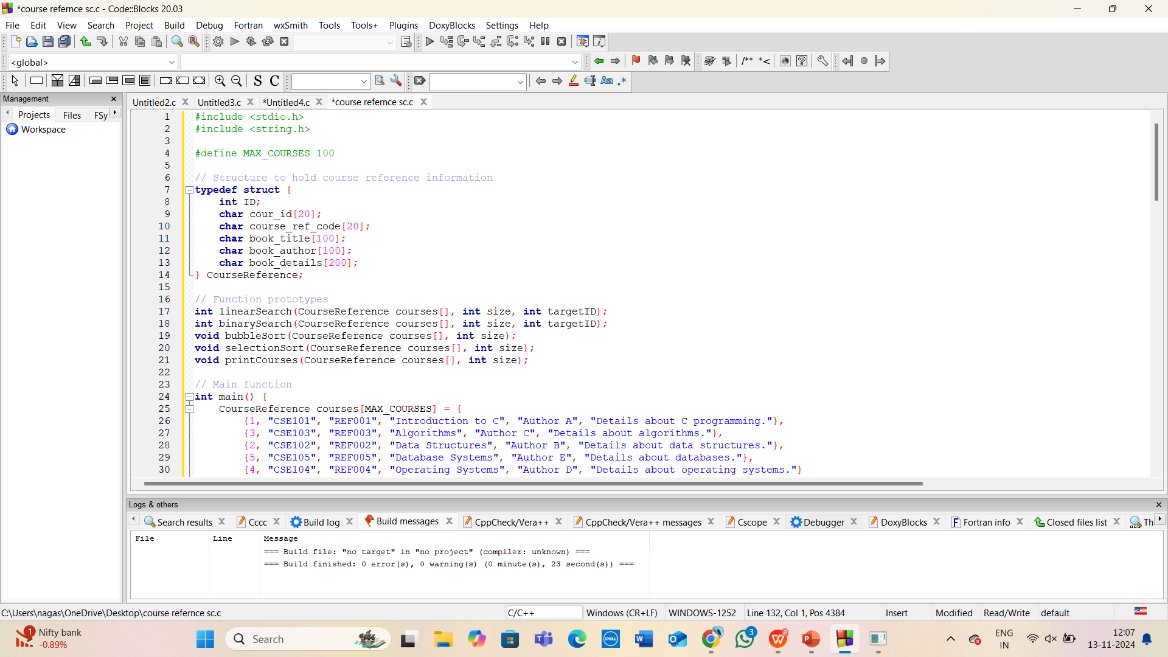
courses[i].ID, courses[i].cour\_id, courses[i].course\_ref\_code,

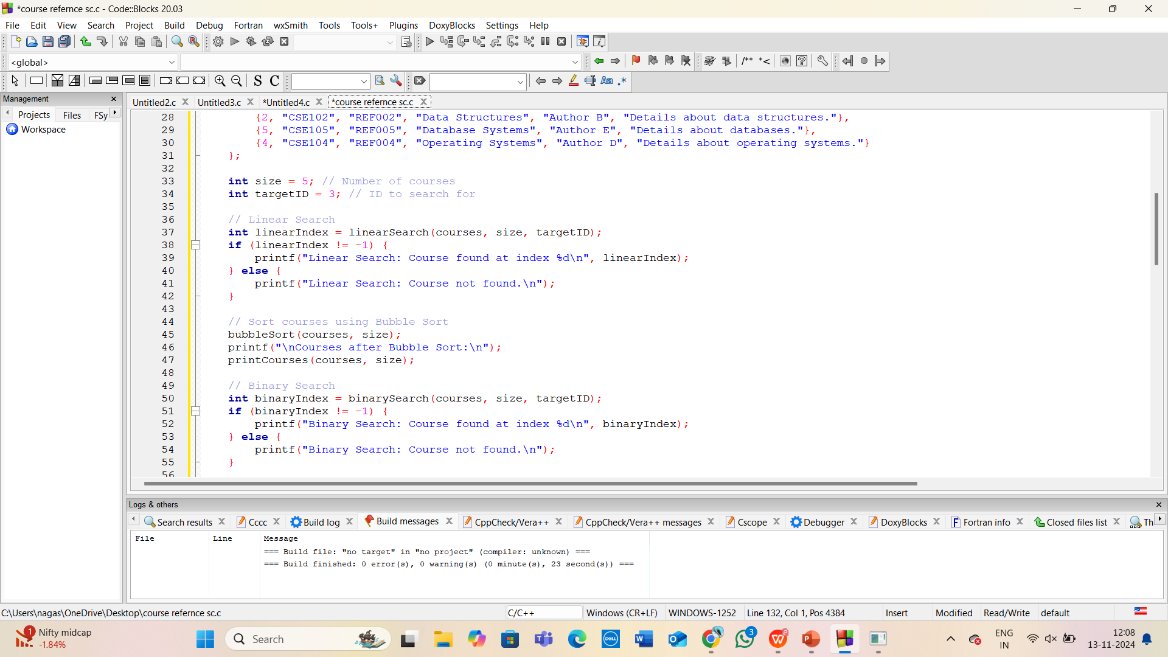
courses[i].book\_title, courses[i].book\_author, courses[i].book\_details);

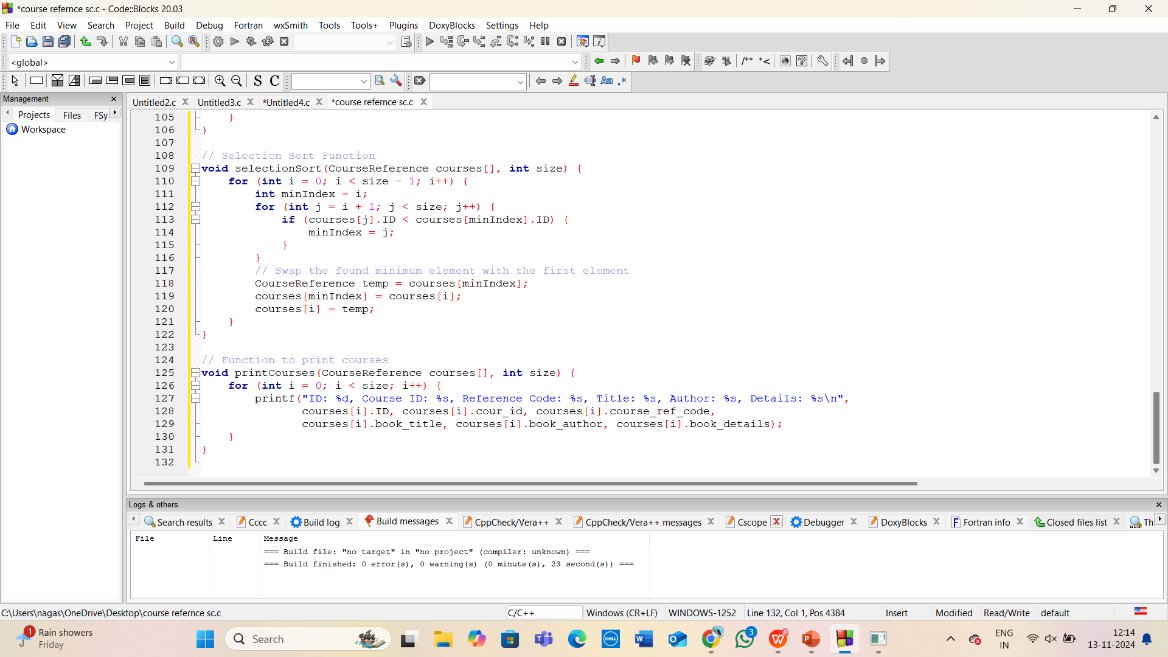
}

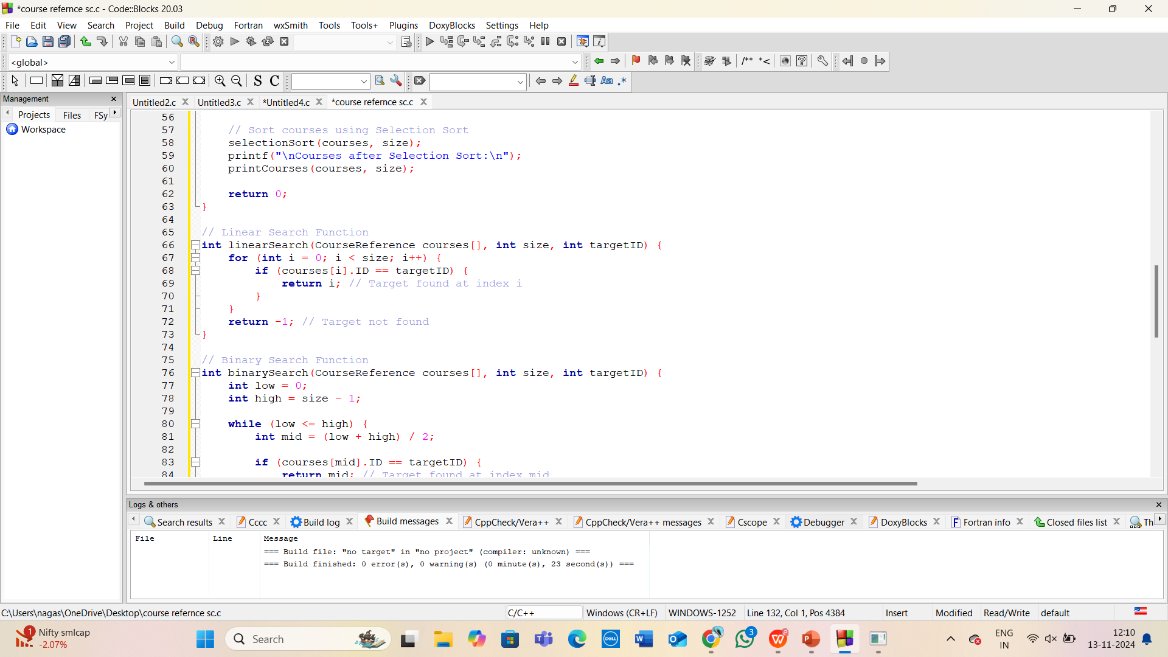
}

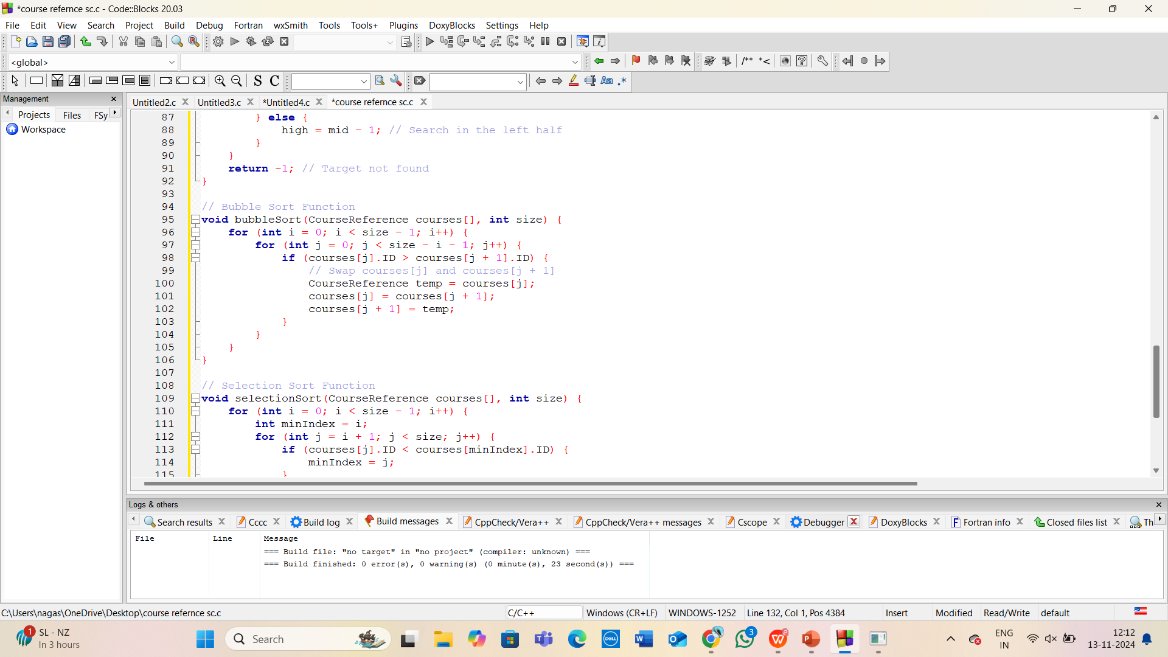
SCREENSHOTS



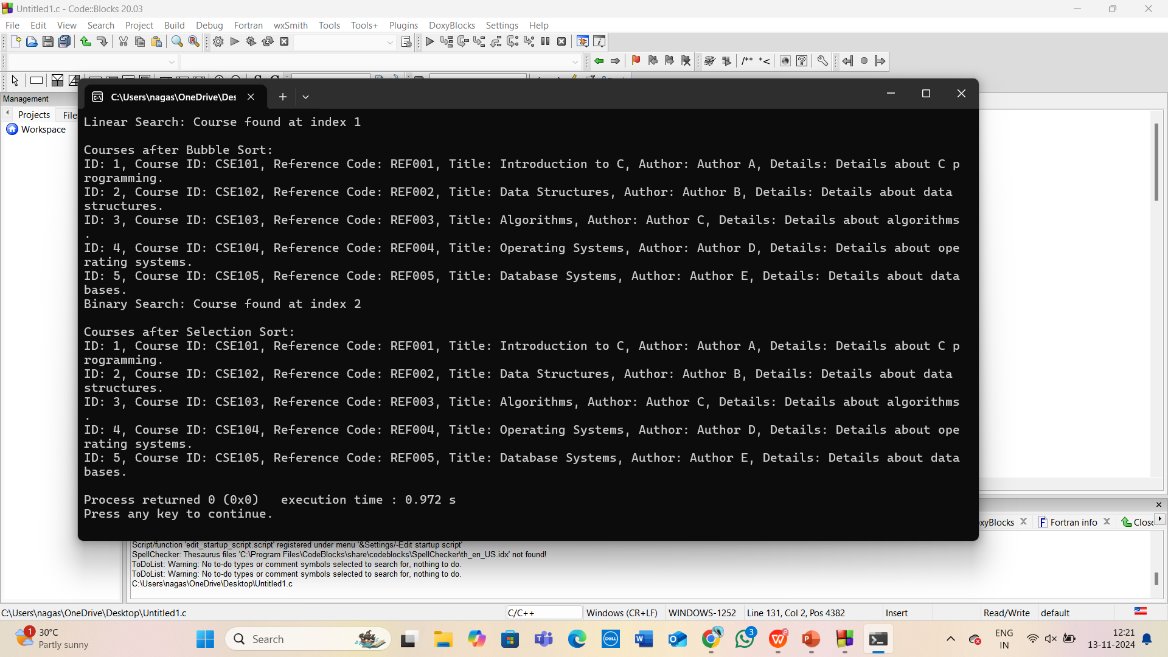








OUTPUT :



Conclusive difference between bubble and selection sort

#include <stdio.h>

#include <string.h>

#define MAX\_ENTRIES 5

typedef struct {

int ID;

int course\_id;

int course\_ref\_code;

char book\_title[100];

char book\_author[100];

char book\_details[200];

} CourseReference;

void bubbleSort(CourseReference data[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (data[j].course\_ref\_code > data[j + 1].course\_ref\_code) {

CourseReference temp = data[j];

data[j] = data[j + 1];

data[j + 1] = temp;

}

}

}

}

void selectionSort(CourseReference data[], int n) {

for (int i = 0; i < n - 1; i++) {

int minIndex = i;

for (int j = i + 1; j < n; j++) {

if (data[j].course\_ref\_code < data[minIndex].course\_ref\_code) {

minIndex = j;

}

}

CourseReference temp = data[minIndex];

data[minIndex] = data[i];

data[i] = temp;

}

}

void printData(CourseReference data[], int n) {

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

printf("ID: %d, Course ID: %d, Course Ref Code: %d, Title: %s, Author: %s, Details: %s\n",

data[i].ID, data[i].course\_id, data[i].course\_ref\_code,

data[i].book\_title, data[i].book\_author, data[i].book\_details);

}

}

int main() {

CourseReference data[MAX\_ENTRIES] = {

{1, 101, 1200, "Data Structures", "Mark Weiss", "C Programming Reference"},

{2, 102, 1350, "Algorithms", "Thomas Cormen", "Comprehensive Algorithm Guide"},

{3, 103, 1100, "Operating Systems", "Abraham Silberschatz", "Modern OS Concepts"},

{4, 104, 1250, "Computer Networks", "Andrew Tanenbaum", "Fundamentals of Networking"},

{5, 105, 1400, "Database Systems", "Ramez Elmasri", "Database Management Systems"}

};

int n = MAX\_ENTRIES;

printf("Original Data:\n");

printData(data, n);

CourseReference bubbleSortedData[MAX\_ENTRIES];

memcpy(bubbleSortedData, data, sizeof(data)); // Copy original data

bubbleSort(bubbleSortedData, n);

printf("\nData after Bubble Sort:\n");

printData(bubbleSortedData, n);

CourseReference selectionSortedData[MAX\_ENTRIES];

memcpy(selectionSortedData, data, sizeof(data)); // Copy original data

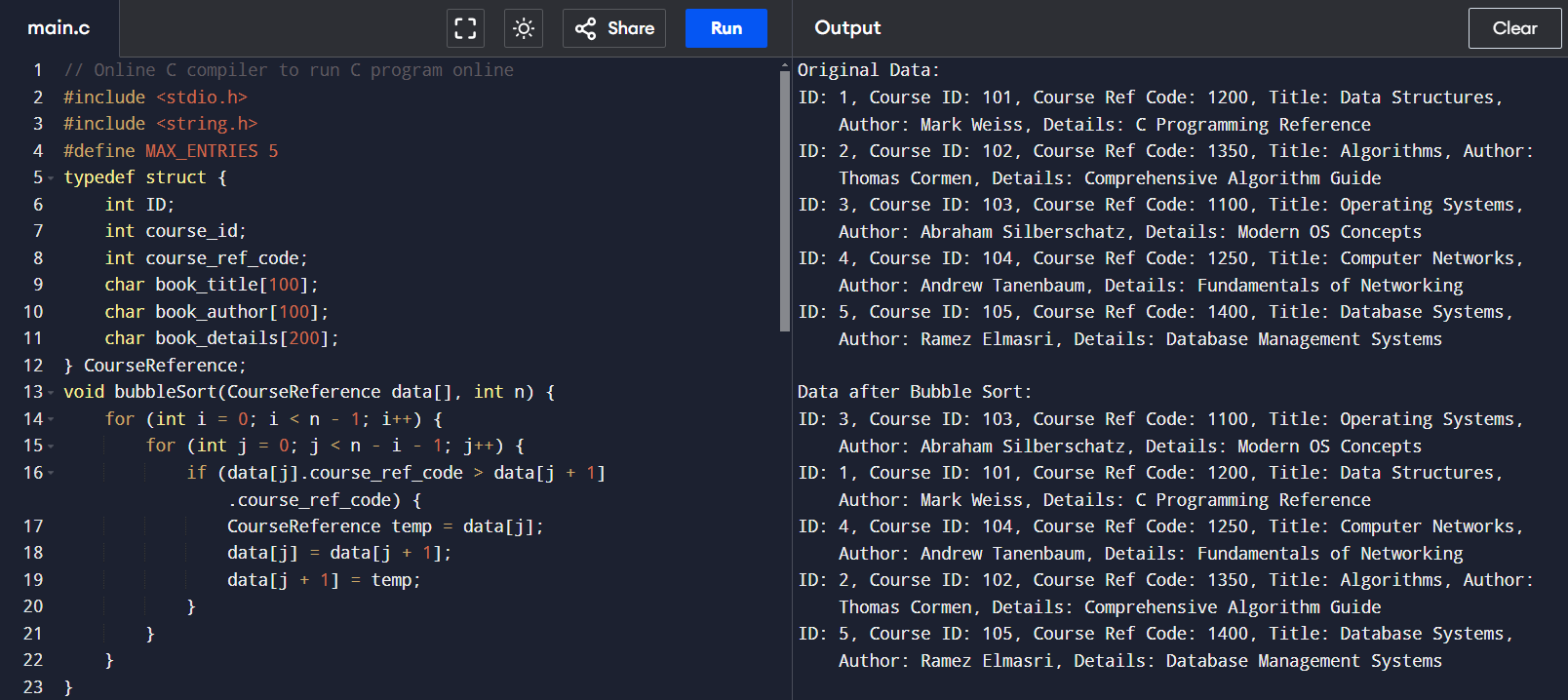
selectionSort(selectionSortedData, n);

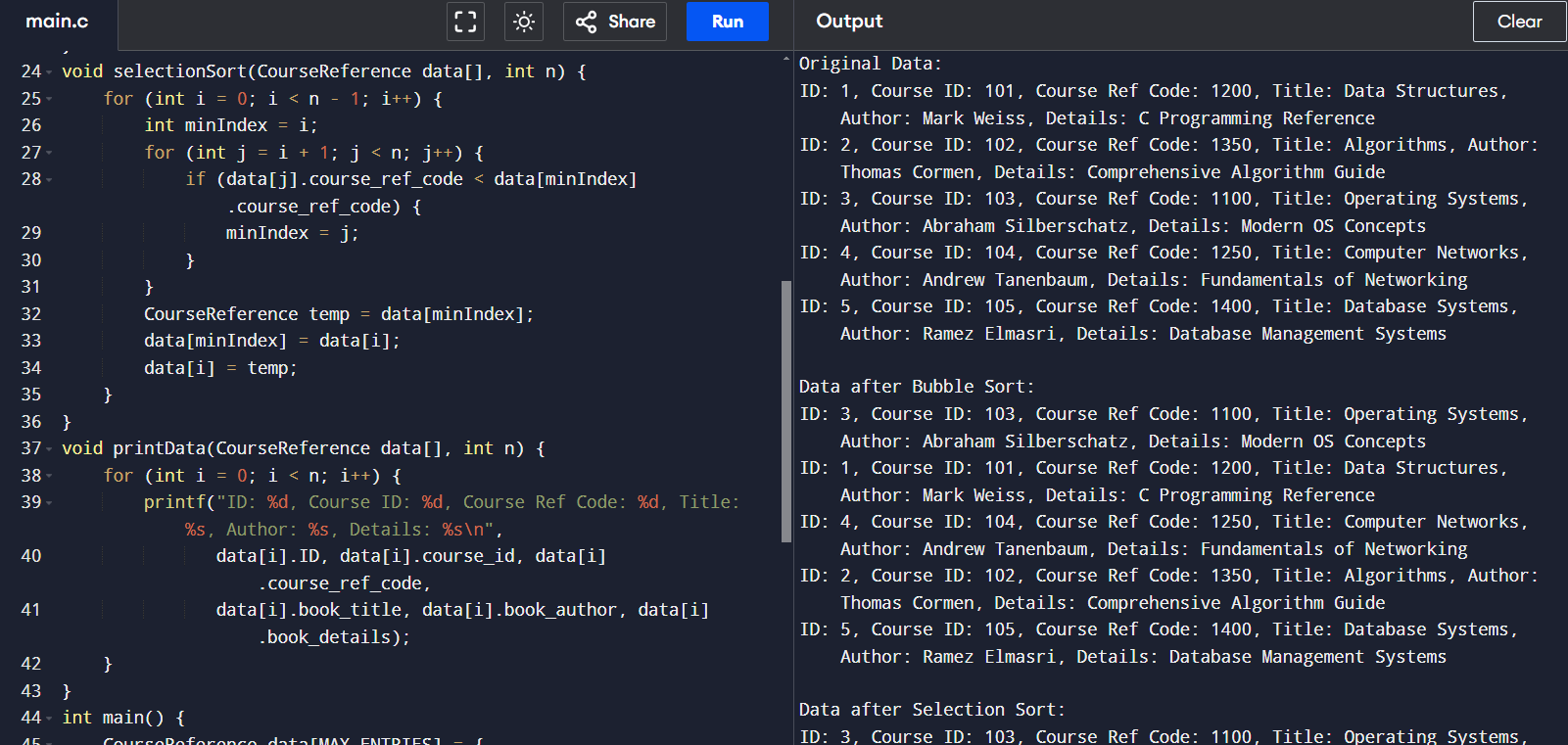
printf("\nData after Selection Sort:\n");

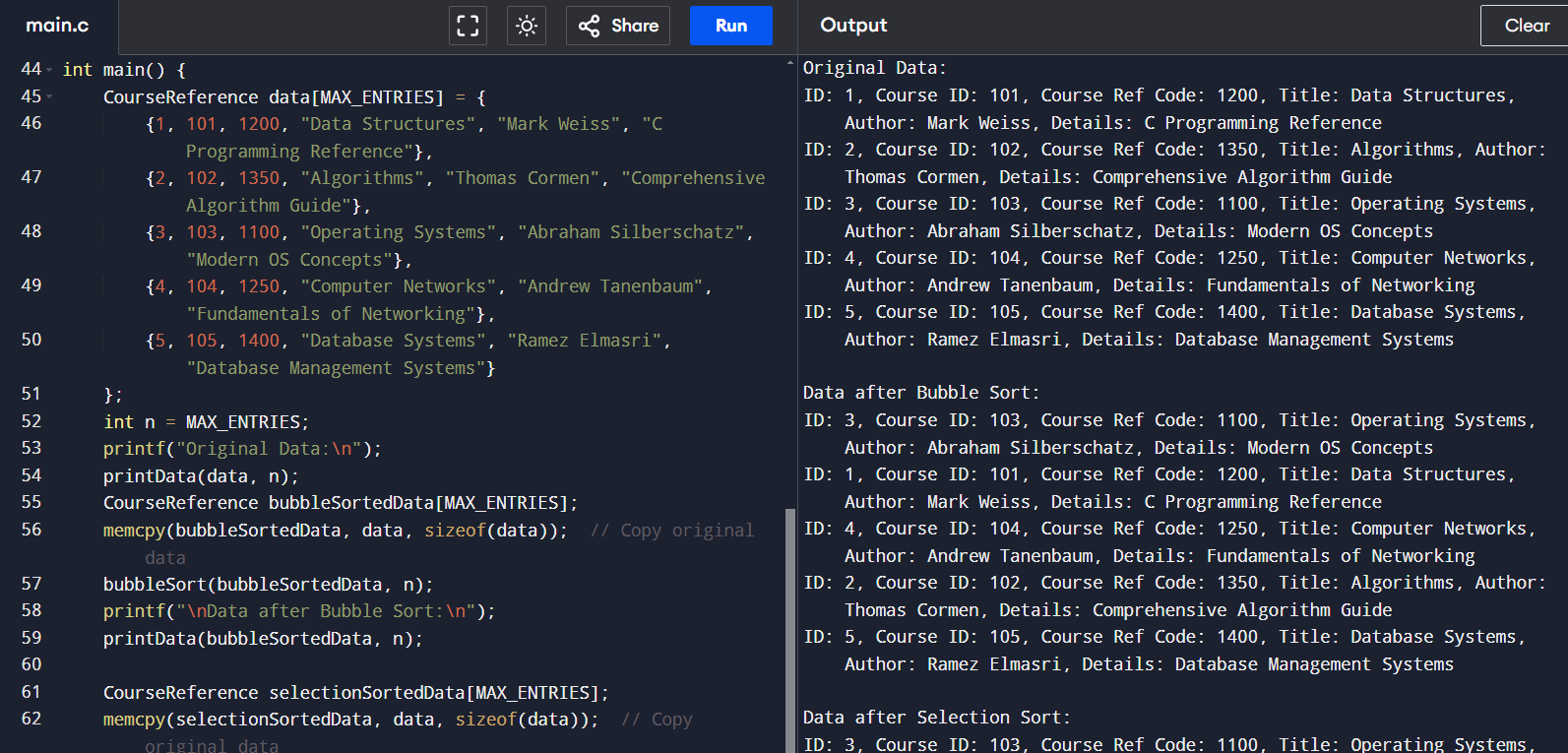
printData(selectionSortedData, n);

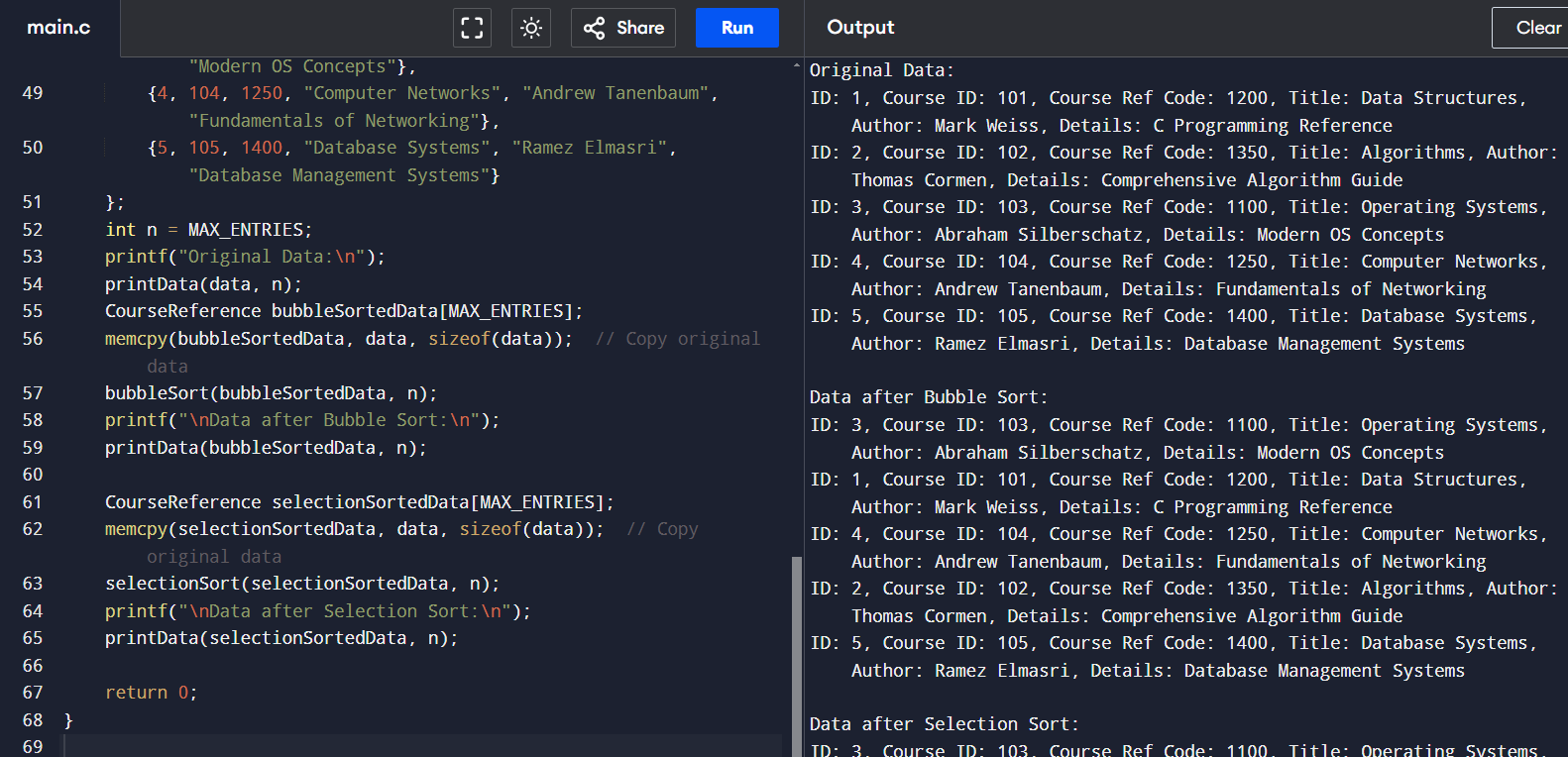
return 0;

}









CONCLUSIVE DIFFERENCE BETWEEN LINEAR AND BINARY SEARCH

#include <stdio.h>

#include <string.h>

#define MAX\_ENTRIES 100

typedef struct {

int ID;

int course\_id;

int course\_ref\_code;

char book\_title[100];

char book\_author[100];

char book\_details[200];

} CourseReference;

int linearSearch(CourseReference data[], int n, int target\_code);

int binarySearch(CourseReference data[], int n, int target\_code);

void sortByCourseRefCode(CourseReference data[], int n);

int main() {

CourseReference data[MAX\_ENTRIES] = {

{1, 101, 1200, "Data Structures", "Mark Weiss", "C Programming Reference"},

{2, 102, 1350, "Algorithms", "Thomas Cormen", "Comprehensive Algorithm Guide"},

{3, 103, 1100, "Operating Systems", "Abraham Silberschatz", "Modern OS Concepts"},

{4, 104, 1250, "Computer Networks", "Andrew Tanenbaum", "Fundamentals of Networking"},

{5, 105, 1400, "Database Systems", "Ramez Elmasri", "Database Management Systems"}

};

int n = 5;

int target\_code;

printf("Enter course reference code to search: ");

scanf("%d", &target\_code);

int index = linearSearch(data, n, target\_code);

if (index != -1) {

printf("Linear Search: Entry found at index %d\n", index);

printf("ID: %d, Course ID: %d, Course Ref Code: %d, Title: %s, Author: %s, Details: %s\n",

data[index].ID, data[index].course\_id, data[index].course\_ref\_code,

data[index].book\_title, data[index].book\_author, data[index].book\_details);

} else {

printf("Linear Search: Entry not found.\n");

}

sortByCourseRefCode(data, n);

index = binarySearch(data, n, target\_code);

if (index != -1) {

printf("Binary Search: Entry found at index %d\n", index);

printf("ID: %d, Course ID: %d, Course Ref Code: %d, Title: %s, Author: %s, Details: %s\n",

data[index].ID, data[index].course\_id, data[index].course\_ref\_code,

data[index].book\_title, data[index].book\_author, data[index].book\_details);

} else {

printf("Binary Search: Entry not found.\n");

}

return 0;

}

int linearSearch(CourseReference data[], int n, int target\_code) {

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

if (data[i].course\_ref\_code == target\_code) {

return i;

}

}

return -1;

}

int binarySearch(CourseReference data[], int n, int target\_code) {

int left = 0, right = n - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (data[mid].course\_ref\_code == target\_code) {

return mid; // Return the index where match is found

} else if (data[mid].course\_ref\_code < target\_code) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return -1;

}

void sortByCourseRefCode(CourseReference data[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (data[j].course\_ref\_code > data[j + 1].course\_ref\_code) {

CourseReference temp = data[j];

data[j] = data[j + 1];

data[j + 1] = temp;

}

}

}

}

