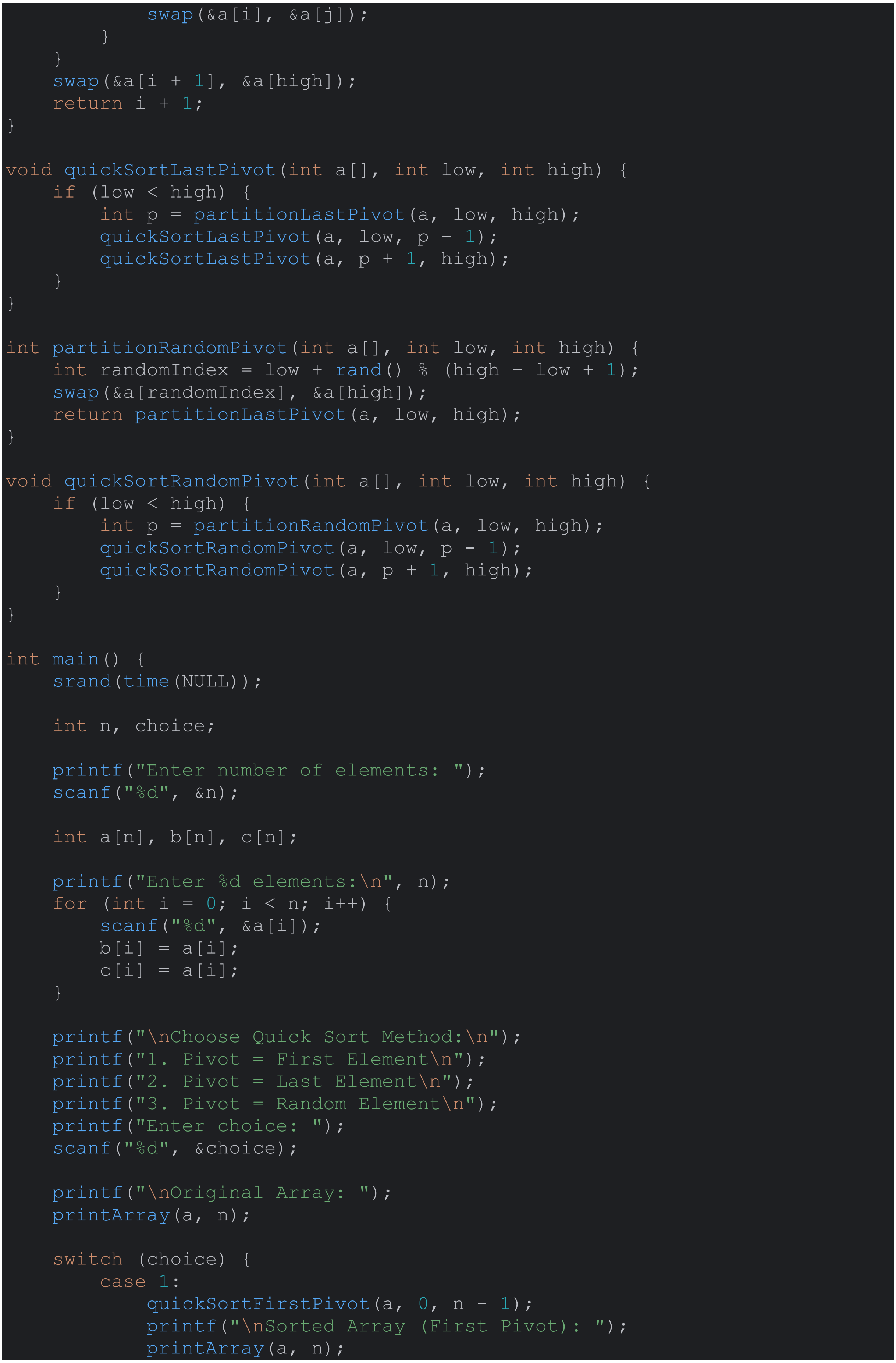
Design Analysis and Algorithm – Lab Work

Week 6

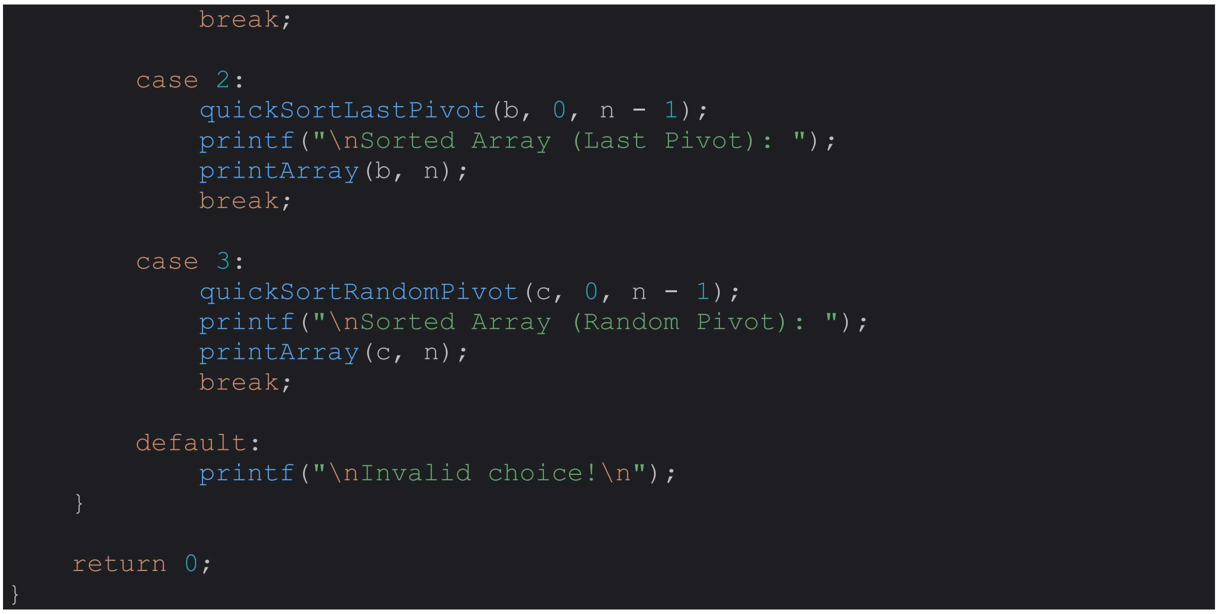
Question 1:Write a Program to perform Quick Sort using first element, middle element and random element.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void printArray(int a[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}

int partitionFirstPivot(int a[], int low, int high) {
    int pivot = a[low];
    int i = low + 1;
    int j = high;

    while (1) {
        while (i <= high && a[i] <= pivot) i++;
        while (a[j] > pivot) j--;

        if (i < j)
            swap(&a[i], &a[j]);
        else
            break;
    }
    swap(&a[low], &a[j]);
    return j;
}

void quickSortFirstPivot(int a[], int low, int high) {
    if (low < high) {
        int p = partitionFirstPivot(a, low, high);
        quickSortFirstPivot(a, low, p - 1);
        quickSortFirstPivot(a, p + 1, high);
    }
}

int partitionLastPivot(int a[], int low, int high) {
    int pivot = a[high];
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (a[j] <= pivot) {
            i++;
```

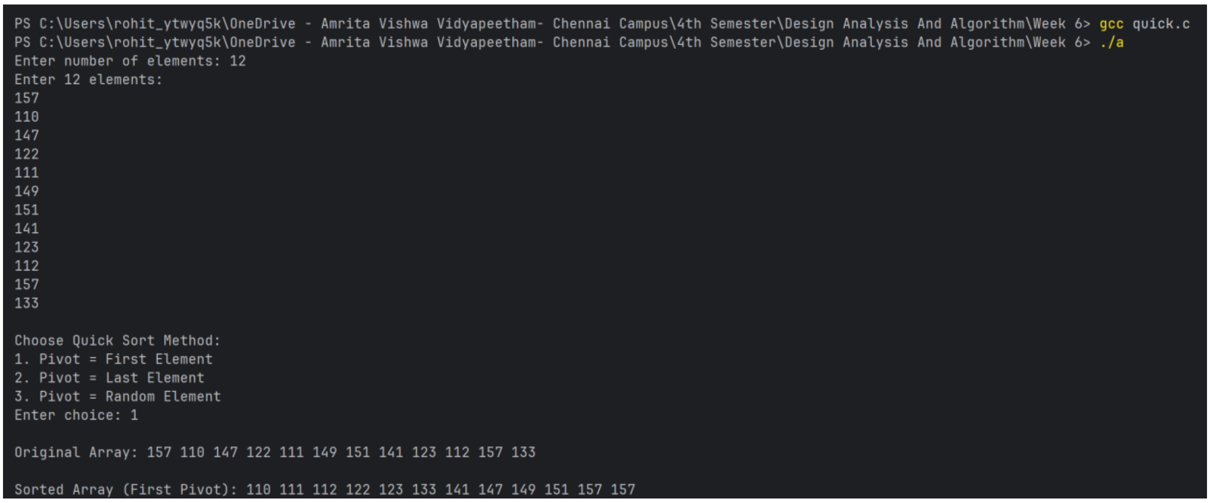


```c
            swap(&a[i], &a[j]);
        }
    }
    swap(&a[i + 1], &a[high]);
    return i + 1;
}

void quickSortLastPivot(int a[], int low, int high) {
    if (low < high) {
        int p = partitionLastPivot(a, low, high);
        quickSortLastPivot(a, low, p - 1);
        quickSortLastPivot(a, p + 1, high);
    }
}

int partitionRandomPivot(int a[], int low, int high) {
    int randomIndex = low + rand() % (high - low + 1);
    swap(&a[randomIndex], &a[high]);
    return partitionLastPivot(a, low, high);
}

void quickSortRandomPivot(int a[], int low, int high) {
    if (low < high) {
        int p = partitionRandomPivot(a, low, high);
        quickSortRandomPivot(a, low, p - 1);
        quickSortRandomPivot(a, p + 1, high);
    }
}

int main() {
    srand(time(NULL));

    int n, choice;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    int a[n], b[n], c[n];

    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
        b[i] = a[i];
        c[i] = a[i];
    }

    printf("\nChoose Quick Sort Method:\n");
    printf("1. Pivot = First Element\n");
    printf("2. Pivot = Last Element\n");
    printf("3. Pivot = Random Element\n");
    printf("Enter choice: ");
    scanf("%d", &choice);

    printf("\nOriginal Array: ");
    printArray(a, n);

    switch (choice) {
        case 1:
            quickSortFirstPivot(a, 0, n - 1);
            printf("\nSorted Array (First Pivot): ");
            printArray(a, n);
```

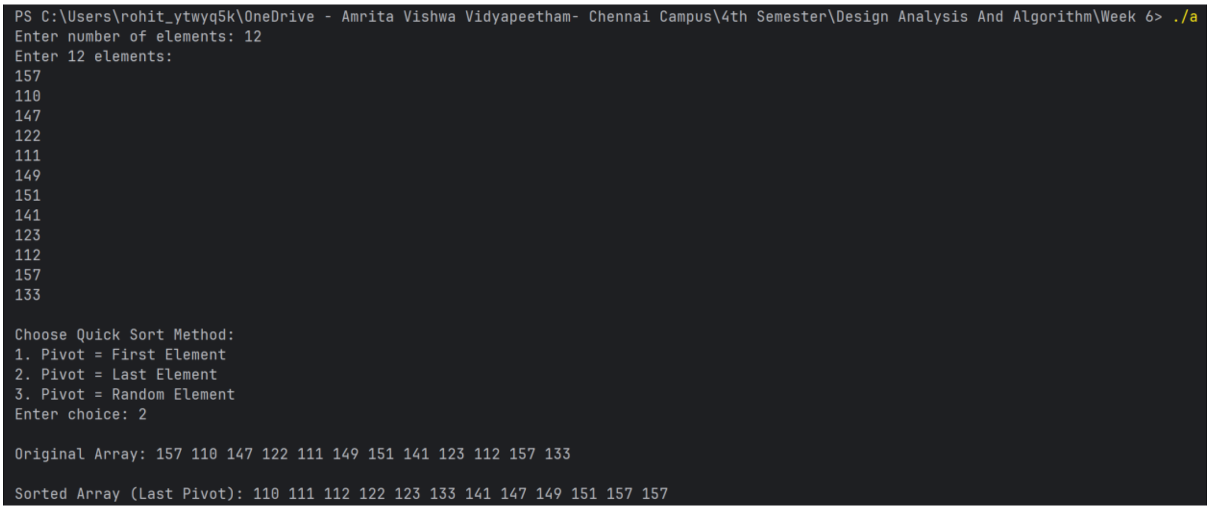


```
            break;

        case 2:
            quickSortLastPivot(b, 0, n - 1);
            printf("\nSorted Array (Last Pivot): ");
            printArray(b, n);
            break;

        case 3:
            quickSortRandomPivot(c, 0, n - 1);
            printf("\nSorted Array (Random Pivot): ");
            printArray(c, n);
            break;

        default:
            printf("\nInvalid choice!\n");
    }

    return 0;
}
```

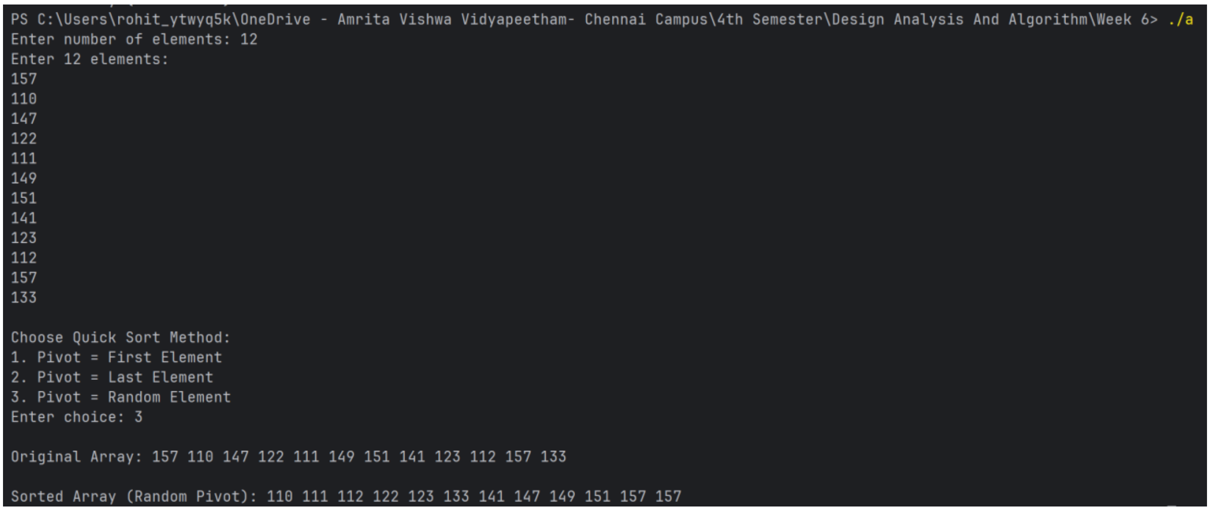
Output:



```
PS C:\Users\rohit_ytwyq5k\OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus\4th Semester\Design Analysis And Algorithm\Week 6> gcc quick.c
PS C:\Users\rohit_ytwyq5k\OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus\4th Semester\Design Analysis And Algorithm\Week 6> ./a
Enter number of elements: 12
Enter 12 elements:
157
110
147
122
111
149
151
141
123
112
157
133

Choose Quick Sort Method:
1. Pivot = First Element
2. Pivot = Last Element
3. Pivot = Random Element
Enter choice: 1

Original Array: 157 110 147 122 111 149 151 141 123 112 157 133

Sorted Array (First Pivot): 110 111 112 122 123 133 141 147 149 151 157 157
```



```
PS C:\Users\rohit_ytwyq5k\OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus\4th Semester\Design Analysis And Algorithm\Week 6> ./a
Enter number of elements: 12
Enter 12 elements:
157
110
147
122
111
149
151
141
123
112
157
133

Choose Quick Sort Method:
1. Pivot = First Element
2. Pivot = Last Element
3. Pivot = Random Element
Enter choice: 2

Original Array: 157 110 147 122 111 149 151 141 123 112 157 133

Sorted Array (Last Pivot): 110 111 112 122 123 133 141 147 149 151 157 157
```



```
PS C:\Users\rohit_ytwyq5k\OneDrive - Amrita Vishwa Vidyapeetham- Chennai Campus\4th Semester\Design Analysis And Algorithm\Week 6> ./a
Enter number of elements: 12
Enter 12 elements:
157
110
147
122
111
149
151
141
123
112
157
133

Choose Quick Sort Method:
1. Pivot = First Element
2. Pivot = Last Element
3. Pivot = Random Element
Enter choice: 3

Original Array: 157 110 147 122 111 149 151 141 123 112 157 133

Sorted Array (Random Pivot): 110 111 112 122 123 133 141 147 149 151 157 157
```

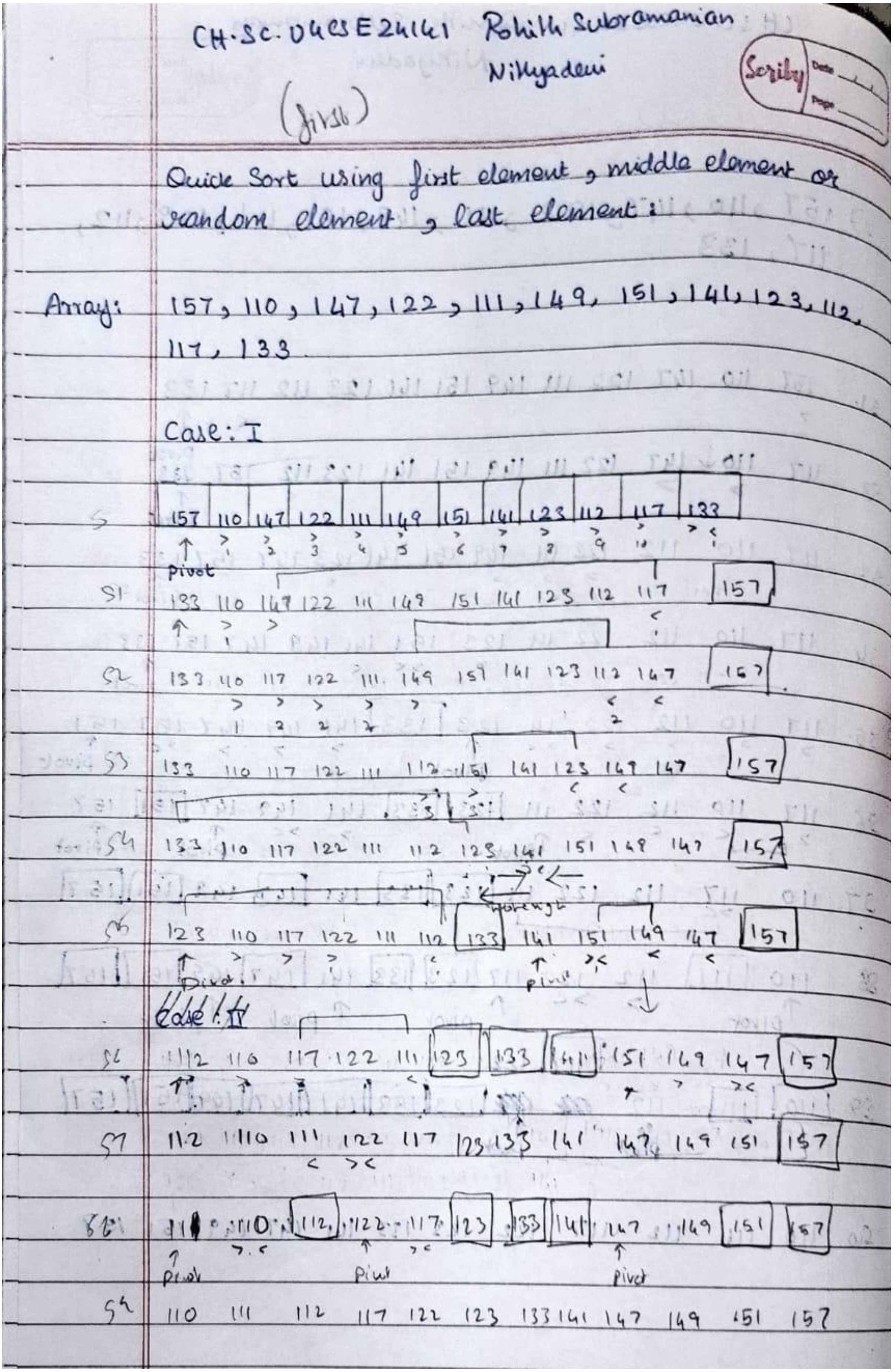
Space Complexity (Worst Case):

In the worst case, the program stores three integer arrays a[n], b[n], and c[n]. Since each integer occupies 4 bytes, the total memory used to store the arrays is 3 × n × 4 = 12n bytes. Apart from this, the program uses a few integer variables such as n, choice, loop variables (i, j), pivot and index variables, and a temporary variable for swapping, which together occupy constant memory. Quick sort is a recursive algorithm, so in the worst case the recursion depth becomes n, and hence the recursion stack occupies O(n) space. Therefore, the worst case Space Complexity of the program is O(n).

Time Complexity (Worst Case):

In the worst case, the pivot chosen (first element pivot or last element pivot, and even random pivot in rare cases) always results in highly unbalanced partitions such that one subarray contains n−1 elements and the other contains 0 elements. In this case, the partition function performs O(n) comparisons for the first call, O(n−1) for the next call, and so on, until O(1). Thus the total number of comparisons becomes n + (n−1) + (n−2) + ... + 1 = n(n−1)/2, which is proportional to n^2. Therefore, the worst case Time Complexity of the program is $O(n^2)$.

Working:



CH.SC.U4CSE24141 Rohith Subramanian Nithyadevi

(first)

Quick Sort using first element, middle element or random element, last element:

Array: 157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133.

Case: I

S 157 | 110 | 147 | 122 | 111 | 149 | 151 | 141 | 123 | 112 | 117 | 133

pivot

S1 133 110 147 122 111 149 151 141 123 112 117 157

S2 133 110 117 122 111 149 151 141 123 112 147 157

S3 133 110 117 122 111 112 151 141 123 149 147 157

S4 133 110 117 122 111 112 123 141 151 149 147 157

wrong

S5 123 110 117 122 111 112 133 141 151 149 147 157

Pivot pivot

Case: II

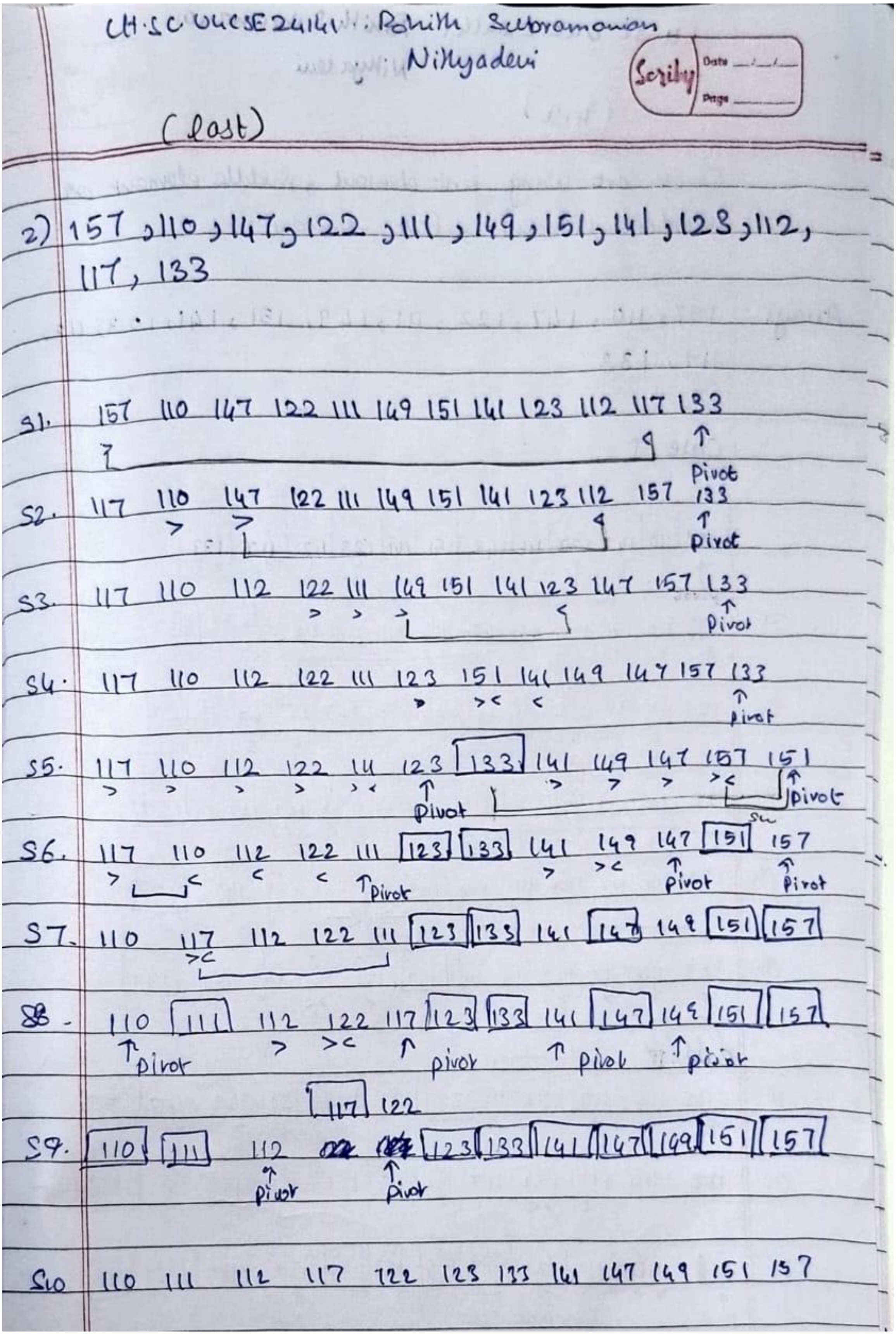
S6 112 110 117 122 111 123 133 141 151 149 147 157

S7 112 110 111 122 117 123 133 141 147 149 151 157

S8 111 110 112 122 117 123 133 141 147 149 151 157

pivot pivot pivot

S9 110 111 112 117 122 123 133 141 147 149 151 157



CH.SC.U4CSE24141 : Rohith Subramanian Nithyadevi

(last)

2) 157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133

S1. 157 110 147 122 111 149 151 141 123 112 117 133

↑ Pivot

S2. 117 110 147 122 111 149 151 141 123 112 157 133

> >

↑ Pivot

S3. 117 110 112 122 111 149 151 141 123 147 157 133

> > >

↑ Pivot

S4. 117 110 112 122 111 123 151 141 149 147 157 133

> >< <

↑ pivot

S5. 117 110 112 122 111 123 $\boxed{133}$ 141 149 147 157 151

> > > > >< ↑ Pivot > > > >< ↑ Pivot

sw

S6. 117 110 112 122 111 $\boxed{123}$ $\boxed{133}$ 141 149 147 $\boxed{151}$ 157

> < < < ↑ Pivot > >< ↑ pivot ↑ Pivot

S7. 110 117 112 122 111 $\boxed{123}$ $\boxed{133}$ 141 $\boxed{147}$ 149 $\boxed{151}$ $\boxed{157}$

><

S8. 110 $\boxed{111}$ 112 122 117 $\boxed{123}$ $\boxed{133}$ 141 $\boxed{147}$ 149 $\boxed{151}$ $\boxed{157}$

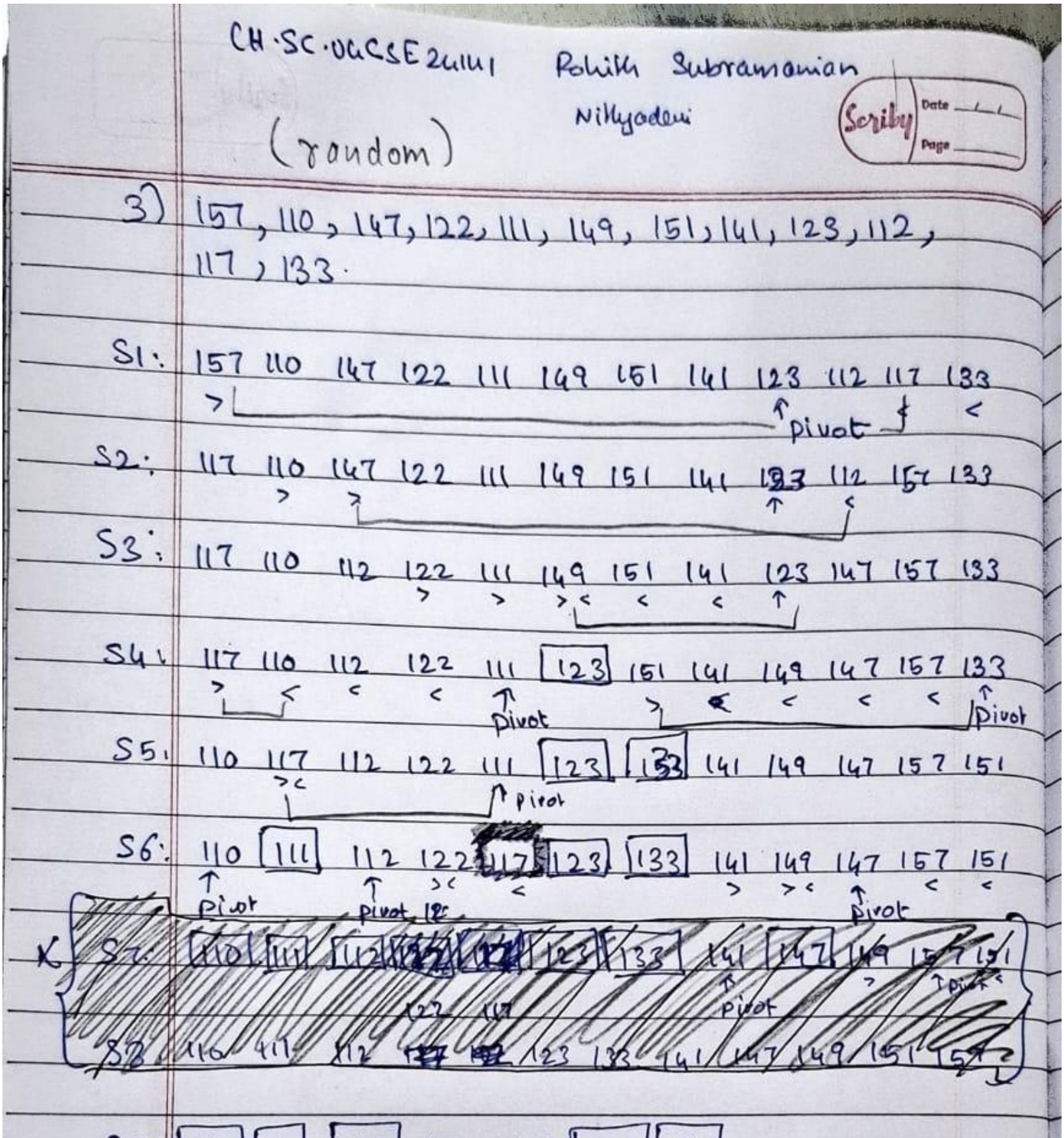
↑ pivot > >< ↑ pivot ↑ pivot ↑ pivot

$\boxed{117}$ 122

S9. $\boxed{110}$ $\boxed{111}$ 112 $\boxed{123}$ $\boxed{133}$ $\boxed{141}$ $\boxed{147}$ $\boxed{149}$ $\boxed{151}$ $\boxed{157}$

↑ pivot ↑ pivot

S10 110 111 112 117 122 123 133 141 147 149 151 157



CH·SC·U4CSE24141 Rohith Subramanian Nithyadevi

(random)

3) 157, 110, 147, 122, 111, 149, 151, 141, 123, 112, 117, 133.

S1: 157 110 147 122 111 149 151 141 123 112 117 133
> ↑pivot <

S2: 117 110 147 122 111 149 151 141 123 112 157 133
> > ↑ <

S3: 117 110 112 122 111 149 151 141 123 147 157 133
> > >< < < ↑

S4: 117 110 112 122 111 123 151 141 149 147 157 133
> < < < ↑pivot > < < < ↑pivot

S5: 110 117 112 122 111 123 133 141 149 147 157 151
>< ↑pivot

S6: 110 111 112 122 117 123 133 141 149 147 157 151
↑pivot ↑pivot >< < > >< ↑pivot < <

~~S7: 110 111 112 122 117 123 133 141 147 149 157 151~~
~~↑pivot > ↑pivot <~~

~~S8 110 111 112 [illegible] [illegible] 123 133 141 147 149 151 157~~

S7: 110 111 112 122 117 123 133 141 149 147 157 151
>< ↑pivot > >< ↑pivot < <

S8: 110 111 112 117 122 123 133 141 147 149 157 151
↑pivot ↑pivot > ↑pivot ><

S9: 110 111 112 117 122 123 133 141 147 149 151 157