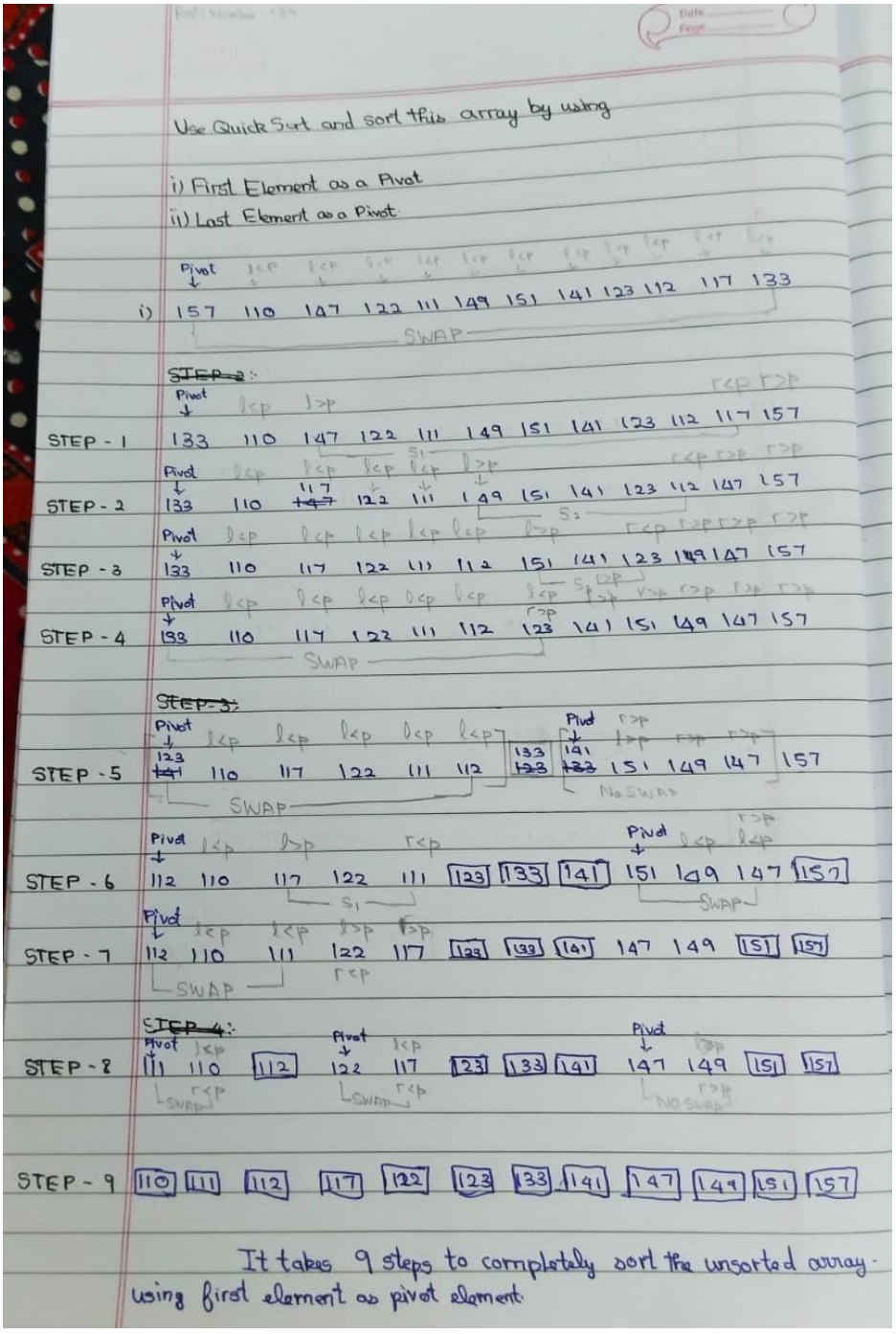
Name : DVN Surya Kiran
Roll no : CH.SC.U4CSE24116
DAA – Quick Sort Using 3 Conditions

I) First Element as Pivot

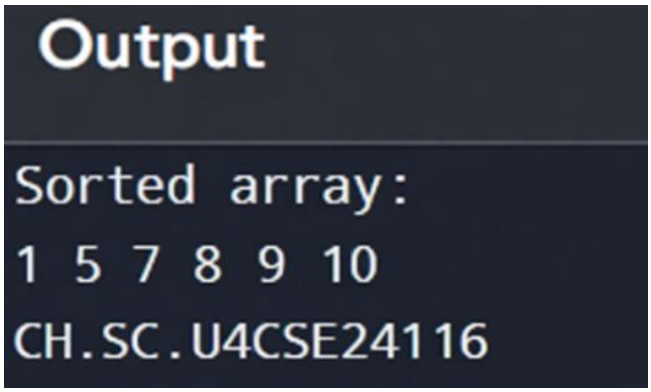
Working :



Program :

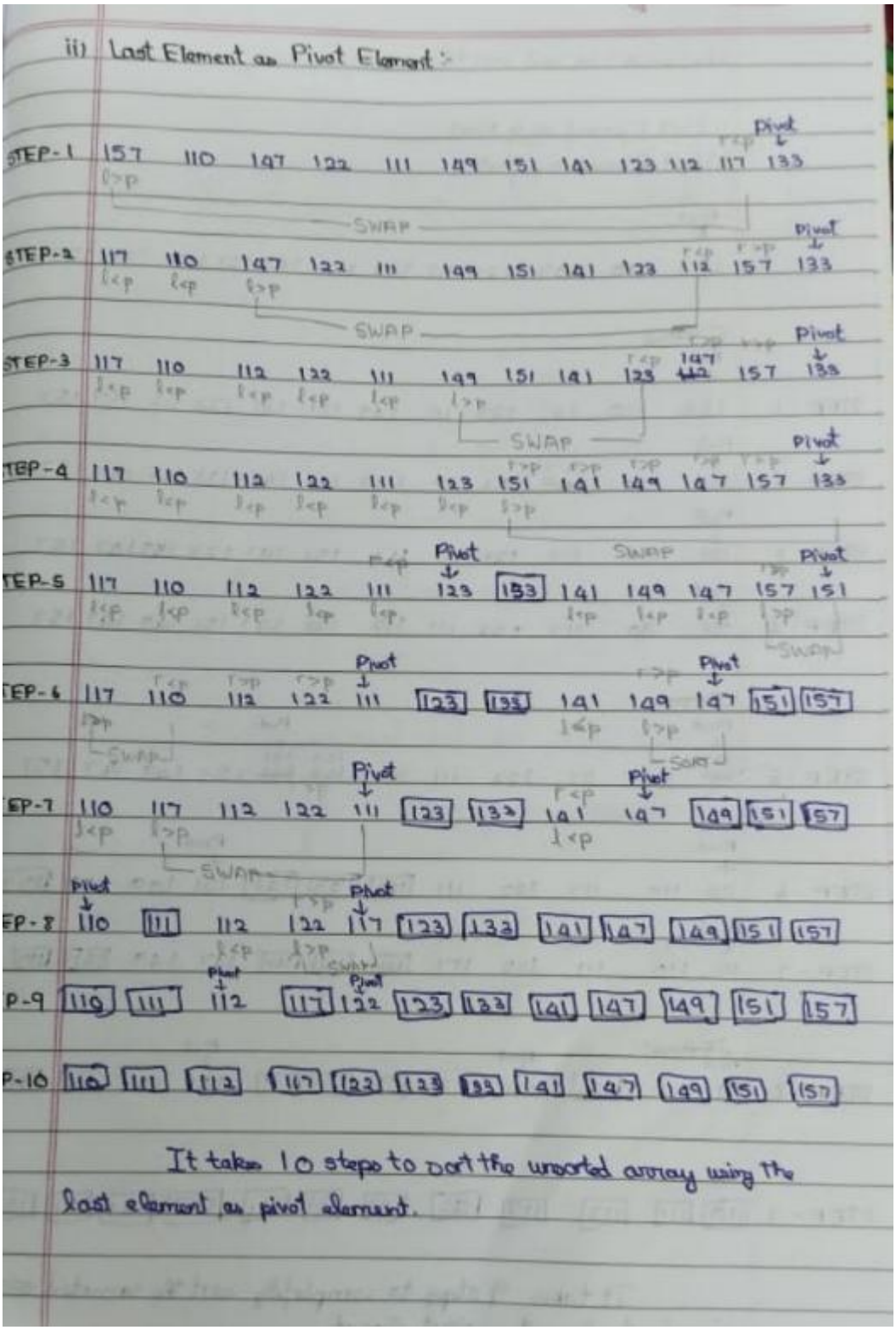
```c
#include <stdio.h>

int partitionFirst(int a[], int low, int high) {
    int pivot = a[low];
    int i = low + 1, j = high, temp;
    while (i <= j) {
        while (i <= high && a[i] <== pivot)
            i++;
        while (a[j] > pivot)
            j--;
        if (i < j) {
            temp = a[j];
            a[i] = a[j];
            a[j] = temp;
        }
    }
    temp = a[low];
    a[low] = a[j];
    a[j] = temp;

}   return j;
}
void quickSortFirst(int a[], int low, int high) {
    if (low < high) {
        int p = partitionFirst(a, low, high);
        quickSortFirst(a, low, p - 1);
        quickSortFirst(a, p + 1, high);
    }
}
int main() {
    int a[] = {10, 7, 8, 9, 1, 5};
    int n = 6, i

    quickSortFirst( a, 0, n - 1);

    printf("Sorted array:\n")
    for (i = 0; i < n, i++)
        printf("%d ", a[i]);
    }
    printf("\nCH.SC.U4CSE24116\n");
```

Output :



II) Last Element as Pivot

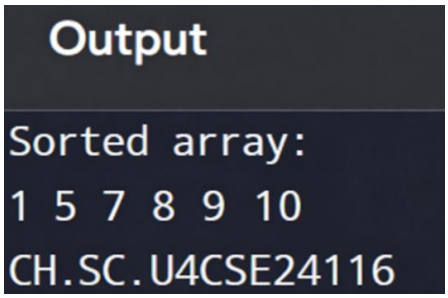
Working :



Program :

```
int partitionLast(int a[], int low, int high) {
    int pivot = a[high];
    int i = low - 1, j, temp;
    for (j = low; j < high; j++) {
        if (a[j] <= pivot) {
            i++;
            temp = a[j];
            a[j] = a[i];
            a[i] = temp;
        }
    }
    temp = a[i + 1];
    a[i + 1] = a[high];
    a[high] = temp;

    return i + 1;
}
void quickSortLast(int a[], int low, int high) {
    if (low < high) {
        int p = partitionLast(a, low, high);
        quickSortLast(a, low, p - 1);
        quickSortLast(a, p + 1, high);
    }
}
int main() {
    int a[] = {10, 7, 8, 9, 1, 5};
    int n = 6, i;
    quickSortLast(a, 0, n - 1);
    printf("Sorted array:\n");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\nCH.SC.U4CSE24116\n");
    return 0;
```

Output :



III) Middle Element as Pivot

Program :

```
2   #include <stdiio.h>
2   #include <stdlib.h>
3   int partitionRandom(int a[], int low, int high) {
4       int random = low + rand() % (high - low + 1);
5       int temp = a[random];
6       a[random] = a[high];
7       a[high] = temp;
8       int pivot = a[high];
9       for (j = low; j < high; j++) {
10          if (a[j] <= pivot) {
11              i++;
12              temp = a[i];
13              a[i] = a[j];
14              a[j] = temp;
15          }
16      }
17      temp = a[i + 1];
18      a[i + 1] = a[high);
19      a[high] = temp;
20      return i + 1;
21  }
22  void quickSortRandom(int a[], inc low, int high) {
23      if (low < high) {
24          int p = partitionRandom(a, low, high);
25          quickSortRandom(a, low, p - 1);
26          quickSortRandom(a, p + 1, high);
27      }
28  }
30  int main() {
30      int a[] = {10, 7, 8, 9, 1, 5};
31      int n - 6, i;
32      quickSortRandom(a, 0, n - 1);
23      printf("Sorted array:\n")
34      for (i = 0, i < n, i++)
```

Output :