QUICK SORT

CODE:

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

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

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

```
while(i <= j){
    while(i <= high && a[i] <= pivot){
        i++;
    }
    while(a[j] > pivot){
        j--;
    }
    if(i < j){
        swap(&a[i], &a[j]);
    }
}
swap(&a[low], &a[j]);
return j;


}
```

int partition Last(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++;
        swap(&a[i], &a[j]);
    }
}
swap(&a[i + 1], &a[high]);
return i + 1;


}
```

int partitionRandom(int a[], int low, int high){ int randomIndex = low + rand() % (high - low + 1); printf("Randomly selected pivot: %d\n", a[randomIndex]); swap(&a[randomIndex], &a[high]); return partitionLast(a, low, high); }

```
void quickSort(int a[], int low, int high, int choice){ if(low < high){ int p;

   if(choice == 1){
        p = partitionFirst(a, low, high);
    }
    else if(choice == 2){
        p = partitionLast(a, low, high);
    }
    else{
        p = partitionRandom(a, low, high);
    }

    quickSort(a, low, p - 1, choice);
    quickSort(a, p + 1, high, choice);
}


}

void copyArray(int src[], int dest[], int n){ for(int i = 0; i < n; i++){ dest[i] = src[i]; } }

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

int main(){ int n, choice;

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

int original[n], temp[n];

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

while(1){
    printf("\n----- QUICK SORT MENU -----\n");
    printf("1. First Element as Pivot\n");
    printf("2. Last Element as Pivot\n");
    printf("3. Random Element as Pivot\n");
    printf("4. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    if(choice == 4){
```

```
        printf("Exiting program...\n");
        break;
    }

    if(choice < 1 || choice > 3){
        printf("Invalid choice! Try again.\n");
        continue;
    }

    copyArray(original, temp, n);

    printf("\nOriginal array:\n");
    display(temp, n);

    quickSort(temp, 0, n - 1, choice);

    printf("Sorted array:\n");
    display(temp, n);
}

return 0;


}
```

OUTPUT:

```
Enter number of elements: 2
Enter elements:
65
45

----- QUICK SORT MENU -----
1. First Element as Pivot
2. Last Element as Pivot
3. Random Element as Pivot
4. Exit
Enter your choice: 1

Original array:
65 45
Sorted array:
45 65

----- QUICK SORT MENU -----
1. First Element as Pivot
2. Last Element as Pivot
3. Random Element as Pivot
4. Exit
Enter your choice: 2
```

```
Enter your choice: 2

Original array:
65 45
Sorted array:
45 65

----- QUICK SORT MENU -----
1. First Element as Pivot
2. Last Element as Pivot
3. Random Element as Pivot
4. Exit
Enter your choice:
3

Original array:
65 45
Randomly selected pivot: 45
Sorted array:
45 65
```

```
----- QUICK SORT MENU -----
1. First Element as Pivot
2. Last Element as Pivot
3. Random Element as Pivot
4. Exit
Enter your choice: 4
Exiting program...
```

NOTES

1)

Quick Sort :-

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

Pivot element = first element

i = left + 1

j = right

Move i right side till A[i] > pivot

Move j left side till A[j] > pivot

If i < j → Swap A[i] & A[j]

If i > j → Swap pivot with A[j]

Index	0	1	2	3	4	5	6	7	8	9	10	11
Array	157	110	147	122	111	149	151	141	123	112	117	133

Step 1:-

Pivot : 157

i = 1, j = 11

No element is greater than 157

j stay at 11

157 > 110 [move i right side]

157 > 147

157 > 122

157 > 133

i moves till end

i = j

So Swap 157 ↔ 133

133	110	147	122	111	149	151	141	123	112	117	157
0 ↑ Pivot	1	2	3	4	5	6	7	8	9	10	11

Step 2:- pivot = 133

i = i j = 10

133 > 110 (i moves right)

133 > 147 (i stays at 144)

Swap (147, 117)

133	110	117	122	111	149	151	141	123	112	147

133 > 122 (moves i right) 112 < 133 (stop at j=9)

133 > 111 (moves i right) i<j

133 < 149 (stop at i: 5) Swap (149, 122)

133	110	117	122	111	112	151	141	123	149	147

112 < 133 (moves i right) 149 > 133 (move, i left)

151 > 133 (stop at i=6) 123 > 133 (stop at j=8)

i > j

Swap (112, 111)

111	110	112	122	117

112 is fixed.

Step 5 :-

Partition (0-1)

i=1, j=1

i = j

Swap (111, 110)

110	111

Sorted

110	111	112	122	117
0	1	2	3	4

Step 6 :- Partition (3-4).

i = 4, j = 4

Pivot = 122

i = j So Swap (122, 117)

117	122

110	111	112	117	122

Sorted

Step 7:- Partition (7-10)

141	151	149	147
7	8	9	10

pivot = 141

157 > 141 (stops at i = 8)

i > j

Swap (pivot, A(j))

(141, 141)

Sorted

141	151	149	147
7	8	9	10

141 fixed.

147 > 141 (j moves left)

149 > 141 (")

151 > 141 (")

149 > 141 (stops at j = 7)

Step 8:- (8-10)

147 < 151 (at j = 10)

Pivot = 151

i = 9, j = 10

149 < 151

147 < 151 (at i = 10)

Swap (151, 147)

147	149	151

Sorted

133	141	147	149	151

finally Sorted array:-

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

2)

Last element as pivot :-

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

Pivot = last (High)

i = low

j = high - 1

move i right while A(i) < pivot (while)

move j left A[j] > pivot (while)

If i<j Swap A[i]; A[j]

If i≥j Swap A[i]; pivot

Step 1:- partition (0-11)

Pivot = 133

i = 0, j = 10

157 > 133 (i stop at i = 0) 117 < 133 (j stop j = 10),

Swap (157 ↔ 117)

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

110 < 133 (move i right)

147 < 133 (stop at i = 2)

157 > 133 (move left)

122 < 133 (stop at j = 9)

Swap (147, 122)

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

112 < 133 (move i right)

111 < 133 (" ")

149 > 133 (stop at i = 5)

147 > 133 (moves left)

123 < 133 (stop at j = 8)

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

123 < 133 (moves right)

150 > 133 (stop at i = 6)

i > j

Swap (pivot, A[i])

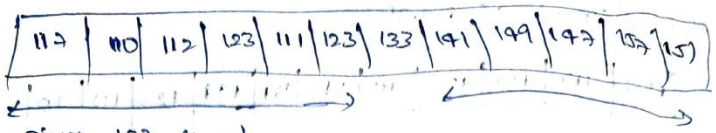
Swap (133, 151)

149 > 133 (move left)

141 > 133 (" ")

151 > 133 (" ")

123 < 133 (stops at j = 5)



Pivot 133 fixed.

Step 2 :- Left (0-5)

Pivot = 123

i = 0 ; j = 4

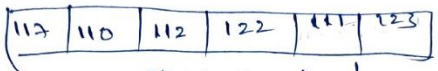
117 < 123 (moves right)

110 < 123 (")

112 < 123 "

111 < 123 "

123 = 123 (stop at i=5)



Pivot 123 fixed.

Step 3:

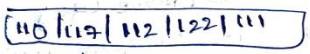
Left (0-4)

Pivot = 111

i = 0 j = 3

117 > 111 (stops)

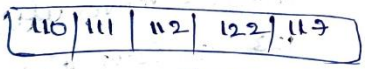
i = 0



i j

117 > 111 (stop at i = 1)

i = j Swap (117, 111)



Pivot 111 is fixed

Step 4 :- (2-4)

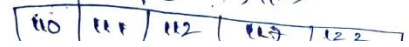
Pivot = 117

i = 2 j = 3

112 < 117 (i moves)

122 > 117 (stops)

j = 2 Swap (117, 112)



Steps : (7-11)

141	149	143	157	151
7	8	9	10	11

Pivot : 151

i = 7 j = 10

141 < 151 (moves right)

149 < 151 "

147 < 151 "

157 < 151 (i stops at i=10)

157 > 151 (j move left)

147 < 151 (j stops)

j = 9

i > j

Swap(157, 151)

141	149	147	151	157

Pivot fixed

Step 6 :- (7-9)

Pivot = 147

i = 7 j = 8

149	149	147

147 < 147 (moves right)

149 > 147 (at j = 8)

Swap (149, 147)

149 > 147 (moves left)

141 < 147 (at i = 7)

141	147	149

Finally sorted array:-

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

3)

iii) Random elements as pivot element

Method / Logic :-

1) Choose random index
2) Swap with first element
3) Use same method used in first element

Sol :-

157	110	147	122	111	149	151	141	123	112	117	133
0	1	2	3	4	5	6	7	8	9	10	11

Step 1:- Take 141 as pivot element

Swap with first element

141	110	147	122	111	149	151	157	123	112	117	133
0	1	2	3	4	5	6	7	8	9	10	11

Pass 1:-

i steps at 147 Swap 147 133.

Steps at 133

141	110	133	122	111	149	131	157	123	112	117	147
0	1	2	3	4	5	6	7	8	9	10	11

Pass 2 :

1 Stop at 149

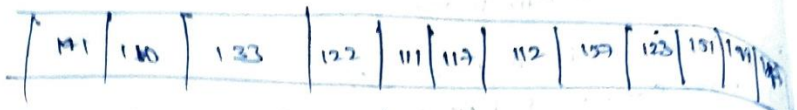
3 Stop at 133. Swap 149 117

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

Pass 3:

i stops at 151 swap 151 ↔ 112

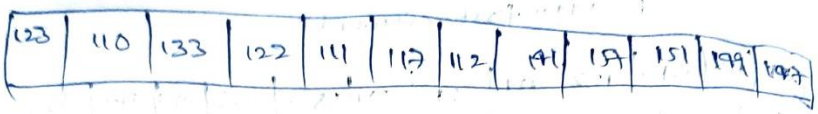
j stops at 112



Pass 4:-

i stop at 151

j stop at 123 swap 151 ↔ 123

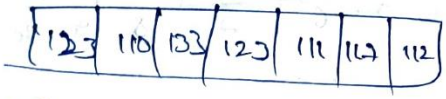


Pivot index = 7

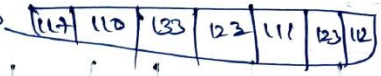
Left subb array [0, 6]

Step 2:- Left Sub arry [0, 6].

Pass 1:-

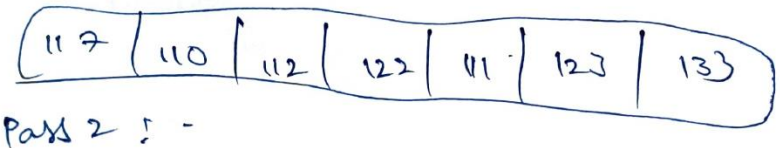


Random pivot = 117

after swap 

i stop at 1

j stop at 6 swap 133 ↔ 112



Pass 2 :-

i stop at 122

j stop at 111 swap 122 ↔ 111

117	110	122	111	112	123	133

Pass 3:

i = 4
s = 3 → 123 → stop

111	110	112	117	122	123	133

Left subarray [0 2]

Step 3:- subarry [0,2) left of 117

Take pivot = 110

Swap with 111

110	111	112

Pass 1:

i = 1 j = 2 → 123 → stop

Already sorted

Right of 117 → already sorted.

Right of 191 → | 157 | 151 | 199 | 147 |

Step 4: subb arry [8,11] | 157 | 151 | 199 | 147 |

Take pivot = 199

Swap with 157

199	151	157	147

Pass 1:

i = 9 Swaps 147 ↔ 151
s = 11

199	147	157	151

Pass 2:-

i = 10 Swap 199 ⟷ 197

j = 9

147	199	157	151

Step 5 :- Right of 149 [10, 11]

157 151

pivot = 131

After Swap 151 157

Sorted

finally

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

At last Random pivot is most efficient that first or last

1. Avoids worst-case performance
2. Give balanced partitions
3. Improve average time complexity
4. better practice performance

P