//**QUESTION 02:** Implement Randomize quick sort and calculate the CPU time.

```
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
int partition(int arr[], int low, int high)
{
int pivot = arr[high];
int i = (low - 1);
for (int j = low; j <= high - 1; j++)
{
if (arr[j] <= pivot) {
i++;
int temp = arr[i];
arr[i]= arr[j];
arr[j] = temp;
}
}
int temp = arr[i+1];
arr[i+1]= arr[high];
arr[high] = temp;
return (i + 1);
}
int partition_r(int arr[], int low, int high)
{
int random = low + rand() % (high - low);
int temp = arr[random];
arr[random]= arr[high];
arr[high] = temp;
return partition(arr, low, high);
}
void quickSort(int arr[], int low, int high)
{
```

```
if (low < high) {
int pi = partition_r(arr, low, high);
quickSort(arr, low, pi - 1);
quickSort(arr, pi + 1, high);
}
}

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

int main()
{
clock_t start_time = clock();
int arr[] = {6,5,4,3,2,1};
int n = sizeof(arr) / sizeof(arr[0]);
quickSort(arr, 0, n - 1);
printf("Sorted array: \n");
printArray(arr, n);
clock_t end_time = clock();
printf("cpu time = %f", double(end_time-start_time));
return 0;
}
```

OUTPUT:

```
┌──(fitsum㉿root)-[~/Desktop/Algorithm]
└─$ ./random_quicksort
Sorted array:
1 2 3 4 5 6 cpu time = 223.000000
```