```
/* /home/student<number>/Examples/Quicksort */
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

#define INPUTN 0
#define INPUTARR 0

void init(int *, int);
void quicksort_parallel(int *, int, int);
void quicksort(int *, int, int);
void print(int *, int);

int main(int *argc, char *argv[]){
	int n = 123, i, *arr;
	if(INPUTN){
		printf("Enter array size: ");
		scanf("%d", &n);
	}
	arr = (int *) malloc(n*sizeof(int));
	if(INPUTARR){
		for(i = 0; i < n; i++){
			printf("a[%d] = ", i);
			scanf("&d", &arr[i]);
		}
	}
	else{
		init(arr, n);
	}

	printf("Unsorted array:\n");
	print(arr, n);
	quicksort_parallel(arr, 0, n);
	printf("*Sorting*\n");
	printf("Sorted array:\n");
	print(arr, n);

	free(arr);
	return 0;
}

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

/* inicializira masiva sys sluchaini stoinosti */
void init(int *arr, int n){
	int i;
	srand(time(NULL));
	/* oboznachavame che razlichnite iteracii na tozi for cikyl shte se izpylnqt paralelno */
	#pragma omp parallel for
	for(i = 0; i < n; i++){
		arr[i] = rand() % (n*2);
	}
}
/* paralelen quicksort */
void quicksort_parallel(int *arr, int left, int right){
	int i = left, j = right, h;
	int x = arr[(left+right)/2];
	do {
		while (arr[i] < x) i++;
		while (arr[j] > x) j--;
		if(i <= j){
			h = arr[i];
			arr[i] = arr[j];
			arr[j] = h;
			i++;
			j--;
		}
	} while(i <= j);
	/* oboznachavame che tozi blok ot programata shte se systoi ot sektori koito mogat da
se izpylnqt paralelno */
	#pragma omp parallel sections
	{
		/* definirame pyrvi sektor koito shte sortira lqvata chast ot masiva*/
		#pragma omp section
		if(left < j) quicksort_parallel(arr, left, j);
		/* definirame vtori sektor koito shte sortira dqsnata chast ot masiva */
		#pragma omp section
		if(i < right) quicksort_parallel(arr, i, right);
	}
}
/* obiknoven quicksort */
void quicksort(int *arr, int left, int right){
	int i = left, j = right, h;
	int x = arr[(left+right)/2];
	do {
		while (arr[i] < x) i++;
		while (arr[j] > x) j--;
		if(i <= j){
			h = arr[i];
			arr[i] = arr[j];
			arr[j] = h;
			i++;
			j--;
		}
	} while(i <= j);
	if(left < j) quicksort(arr, left, j);
	if(i < right) quicksort(arr, i, right);
}
```