**Lab Description:**

\*\*Lab description not on write up

\*In README file

**Graph:**

**Main:**

#include <iostream> // cout, endl

#include <stdio.h> // rand, srand

#include <time.h> // time

#include <chrono> // using chrono and time clocking algorithms

void quickSort(int\*, int, int); // sorts array using Quick Sort algorithm

int partition(int\*, int, int); // partitions array for quickSort Function

void swap(int&, int&); // swaps two integers

void print(int\*, int); // prints array

int\* makeArray(int); // creates a dynamically allocated array with passed limit

int main() {

srand(time(NULL));

int arr[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1}; // Hard Coded Examples and edge cases

int arr2[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int arr3[] = {1, 3, 2, 5, 4, 7, 6, 9, 8, 10};

std::cout << "Unsorted: " << std::endl;

std::cout << "Array 1:" << std::endl;

print(arr, 10);

std::cout << "Array 2:" << std::endl;

print(arr2, 10);

std::cout << "Array 3:" << std::endl;

print(arr3, 10);

quickSort(arr, 0, 10);

quickSort(arr2, 0, 10);

quickSort(arr3, 0, 10);

std::cout << "Sorted: " << std::endl;

std::cout << "Array 1:" << std::endl;

print(arr, 10);

std::cout << "Array 2:" << std::endl;

print(arr2, 10);

std::cout << "Array 3:" << std::endl;

print(arr3, 10);

int limitArray[] = {100, 500, 1500, 5000, 10000, 25000, 50000, 75000, 100000, 250000, 500000, 800000, 1000000, 1500000, 5000000};

int limitSize = 15;

std::cout << "\nStart of Large Case Arrays:" << std::endl; // Random examples

for (int i = 0; i < limitSize; i++) {

int\* a = makeArray(limitArray[i]);

std::cout << "Starting Quick Sort of array size " << limitArray[i] << ":" << std::endl;

auto start = std::chrono::system\_clock::now();

quickSort(a, 0, limitArray[i]);

auto end = std::chrono::system\_clock::now();

std::chrono::duration<double> elapsed\_seconds = end - start;

std::time\_t end\_time = std::chrono::system\_clock::to\_time\_t(end);

std::cout << "Finished at " << std::ctime(&end\_time) << "Elapsed time: " << elapsed\_seconds.count() << "s\n";

delete [] a;

}

return 0;

}

/\*

\* makeArray Function:

\* Creates and fills an array based off of a specified limit

\*/

int\* makeArray(int len) {

int\* rtn = new int[len];

for (int i = 0; i < len; i++) {

rtn[i] = 1 + rand() % 100;

}

return rtn;

}

/\*

\* quickSort Function:

\* Uses quick sort algorithm to sort an array of integers

\*/

void quickSort(int\* arr, int start, int end) {

if (start < end) {

int p = partition(arr, start, end);

quickSort(arr, start, p);

quickSort(arr, p + 1, end);

}

}

/\*

\* partition Function:

\* Returns a partition for the quickSort function

\*/

int partition(int\* arr, int start, int end) {

int p = arr[end - 1];

int i = start - 1;

for (int j = start; j <= end - 2; j++) {

if (arr[j] < p) {

i++;

swap(arr[j], arr[i]);

}

}

swap(arr[end - 1], arr[i + 1]);

return i + 1;

}

/\*

\* swap Function:

\* Swaps two integers

\*/

void swap(int &val1, int &val2) {

int temp = val1;

val1 = val2;

val2 = temp;

}

/\*

\* print Function:

\* Prints the array for hard coded

\*/

void print(int\* arr, int max) {

std::cout << "===========================" << std::endl;

for (int i = 0; i < max; i++) {

std::cout << arr[i] << " ";

}

std::cout << std::endl;

}

**Sample Output:**

Unsorted:

Array 1:

===========================

10 9 8 7 6 5 4 3 2 1

Array 2:

===========================

1 2 3 4 5 6 7 8 9 10

Array 3:

===========================

1 3 2 5 4 7 6 9 8 10

Sorted:

Array 1:

===========================

1 2 3 4 5 6 7 8 9 10

Array 2:

===========================

1 2 3 4 5 6 7 8 9 10

Array 3:

===========================

1 2 3 4 5 6 7 8 9 10

Start of Large Case Arrays:

Starting Quick Sort of array size 100:

Finished at Tue Oct 29 10:17:13 2019

Elapsed time: 1.0706e-05s

Starting Quick Sort of array size 500:

Finished at Tue Oct 29 10:17:13 2019

Elapsed time: 5.7763e-05s

Starting Quick Sort of array size 1500:

Finished at Tue Oct 29 10:17:13 2019

Elapsed time: 0.000191986s

Starting Quick Sort of array size 5000:

Finished at Tue Oct 29 10:17:13 2019

Elapsed time: 0.000909656s

Starting Quick Sort of array size 10000:

Finished at Tue Oct 29 10:17:13 2019

Elapsed time: 0.00255494s

Starting Quick Sort of array size 25000:

Finished at Tue Oct 29 10:17:13 2019

Elapsed time: 0.0206038s

Starting Quick Sort of array size 50000:

Finished at Tue Oct 29 10:17:13 2019

Elapsed time: 0.0691004s

Starting Quick Sort of array size 75000:

Finished at Tue Oct 29 10:17:14 2019

Elapsed time: 0.152605s

Starting Quick Sort of array size 100000:

Finished at Tue Oct 29 10:17:14 2019

Elapsed time: 0.245582s

Starting Quick Sort of array size 250000:

Finished at Tue Oct 29 10:17:15 2019

Elapsed time: 0.90042s

Starting Quick Sort of array size 500000:

Finished at Tue Oct 29 10:17:18 2019

Elapsed time: 3.66271s

Starting Quick Sort of array size 800000:

Finished at Tue Oct 29 10:17:42 2019

Elapsed time: 23.8436s

Starting Quick Sort of array size 1000000:

Finished at Tue Oct 29 10:17:58 2019

Elapsed time: 27.1275s

Starting Quick Sort of array size 1500000:

Finished at Tue Oct 29 10:18:31 2019

Elapsed time: 32.6925s

Starting Quick Sort of array size 5000000:

Finished at Tue Oct 29 10:23:49 2019

Elapsed time: 317.029s