#include <stdio.h>

#include <stdlib.h>

#include <time.h>

float generateRandomNumber(float min, float max) {

return min + ((float)rand() / RAND\_MAX) \* (max - min);

}

int compare(const void\* a, const void\* b) {

float fa = \*(const float\*)a;

float fb = \*(const float\*)b;

if (fa < fb) return -1;

else if (fa > fb) return 1;

else return 0;

}

int main() {

srand(time(NULL));

int inp;

printf("Enter input size: ");

scanf("%d", &inp);

if (inp <= 0 || inp > 1000000) {

printf("Invalid input size. Exiting.\n");

return -1;

}

float\* temp = (float\*)malloc(sizeof(float) \* inp);

float\* pressure = (float\*)malloc(sizeof(float) \* inp);

if (temp == NULL || pressure == NULL) {

printf("Memory allocation failed. Exiting.\n");

return -1;

}

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

temp[i] = generateRandomNumber(-20, 50);

pressure[i] = generateRandomNumber(950, 1050);

}

clock\_t start\_time, end\_time;

float maxTemp, minPressure;

printf("Linear Search \n");

maxTemp = temp[0];

start\_time = clock();

for (int j = 1; j < inp; j++) {

if (temp[j] > maxTemp) {

maxTemp = temp[j];

}

}

end\_time = clock();

printf("Linear Search - Maximum Temperature: %.2f\n", maxTemp);

printf("Time Taken: %.3f ms\n", ((double)(end\_time - start\_time) / CLOCKS\_PER\_SEC) \* 1000);

minPressure = pressure[0];

start\_time = clock();

for (int j = 1; j < inp; j++) {

if (pressure[j] < minPressure) {

minPressure = pressure[j];

}

}

end\_time = clock();

printf("Linear Search - Minimum Pressure: %.2f\n", minPressure);

printf("Time Taken: %.3f ms\n\n", ((double)(end\_time - start\_time) / CLOCKS\_PER\_SEC) \* 1000);

printf("Quadratic Search \n");

maxTemp = temp[0];

start\_time = clock();

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

int isMax = 1;

for (int j = 0; j < inp; j++) {

if (temp[i] < temp[j]) {

isMax = 0;

break;

}

}

if (isMax) {

maxTemp = temp[i];

break;

}

}

end\_time = clock();

printf("Quadratic Search - Maximum Temperature: %.2f\n", maxTemp);

printf("Time Taken: %.3f ms\n", ((double)(end\_time - start\_time) / CLOCKS\_PER\_SEC) \* 1000);

minPressure = pressure[0];

start\_time = clock();

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

int isMin = 1;

for (int j = 0; j < inp; j++) {

if (pressure[i] > pressure[j]) {

isMin = 0;

break;

}

}

if (isMin) {

minPressure = pressure[i];

break;

}

}

end\_time = clock();

printf("Quadratic Search - Minimum Pressure: %.2f\n", minPressure);

printf("Time Taken: %.3f ms\n\n", ((double)(end\_time - start\_time) / CLOCKS\_PER\_SEC) \* 1000);

printf("Search for First Temperature -\n");

float\* sortedTemp = (float\*)malloc(sizeof(float) \* inp);

if (sortedTemp == NULL) {

printf("Memory allocation failed. Exiting.\n");

free(temp);

free(pressure);

return -1;

}

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

sortedTemp[i] = generateRandomNumber(20, 50);

}

qsort(sortedTemp, inp, sizeof(float), compare);

int index = -1;

start\_time = clock();

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

if (sortedTemp[i] >= 30) {

index = i;

break;

}

}

end\_time = clock();

printf("Linear Search %d\n", index);

printf("Time Taken: %.3f ms\n", ((double)(end\_time - start\_time) / CLOCKS\_PER\_SEC) \* 1000);

int left = 0, right = inp - 1;

index = -1;

start\_time = clock();

while (left <= right) {

int mid = left + (right - left) / 2;

if (sortedTemp[mid] >= 30) {

index = mid;

right = mid - 1;

} else {

left = mid + 1;

}

}

end\_time = clock();

printf("Binary Search %d\n", index);

printf("Time Taken: %.3f ms\n\n", ((double)(end\_time - start\_time) / CLOCKS\_PER\_SEC) \* 1000);

free(temp);

free(pressure);

free(sortedTemp);

return 0;

}