**Algorithm Lab. Class Assignment-2**

**CSE Group 1**

**Date: - 16th July 2021**

**Name-Ritam Hudait,**

**Roll-1905797**

1. Write a program that takes three variables (A, B, C) as separate parameters and rotates the values stored so that value A goes to B, B to C, and C to A by using SWAP(x,y) as a function that swaps/exchanges the numbers x & y.

Program::

#include<stdio.h>

void swap(int \*x,int \*y){

int temp = \*x;

\*x = \*y;

\*y = temp;

}

int main() {

printf("enter three variables \n");

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

swap(&a,&b);

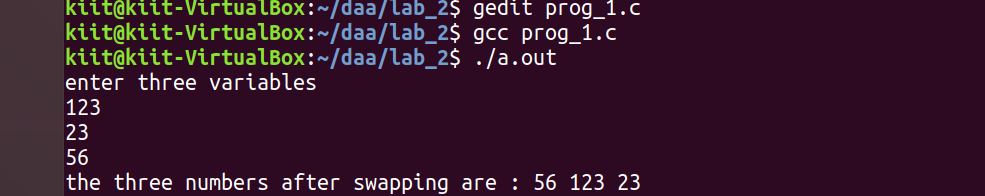
swap(&a,&c);

printf("the three numbers after swapping are : %d %d %d\n",a,b,c);

return 0;

}

Output::



2.Let A be n\*n square matrix array. WAP by using appropriate user-defined functions for the following:

* 1. Find the number of nonzero elements in A
  2. Find the sum of the elements above the leading diagonal.
  3. Display the elements below the minor diagonal.
  4. Find the product of the diagonal elements.

Program::

#include <stdio.h>

int main()

{

int n;

scanf("%d", &n);

int z = 0;

int s = 0;

int p = 1;

int matrix[n][n];

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

scanf("%d", &matrix[i][j]);

if (matrix[i][j]) {

z++;

}

if (i == j){

p \*=matrix[i][j];

}

if (j > i){

s += matrix[i][j];

}

}

}

printf("the number of non zero elements : %d\n",z);

printf("the sum of elements above diagonal: %d\n", s);

printf("the product of diagonal elements: %d\n", p);

printf("the minor diagonal elements: \n");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

if (i + j >= n){

printf("%d, ",matrix[i][j]);

}

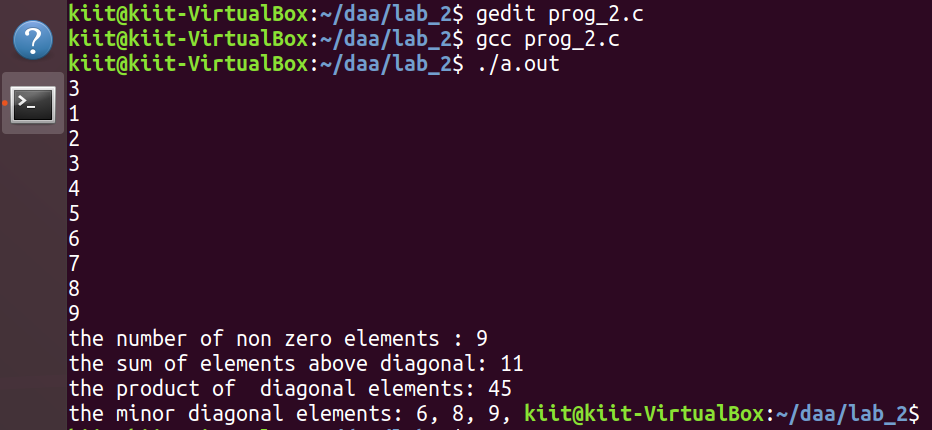
}

}

return 0;

}

Ouput:



3.WAP in C to store 1 million integers in an array. To search an element in that array and find out its time complexity (best, worst, and average).

Program::

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int main() {

int n =30000;

int a[n];

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

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

a[i] = i + 1;

}

int best = a[0];

int worst = a[n - 1];

int avg = a[n / 2];

time\_t strt, end;

strt = clock();

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

if (best == a[i]) {

end = clock();

double t = end - strt;

printf("best case time: %f\n", (t / CLOCKS\_PER\_SEC));

break;

}

}

strt = clock();

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

if (avg == a[i]) {

end = clock();

double t = end - strt;

printf("avg case time: %f\n", (t / CLOCKS\_PER\_SEC));

break;

}

}

strt = clock();

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

if (worst == a[i]) {

end = clock();

double t = end - strt;

printf("worst case time: %f\n", (t / CLOCKS\_PER\_SEC));

break;

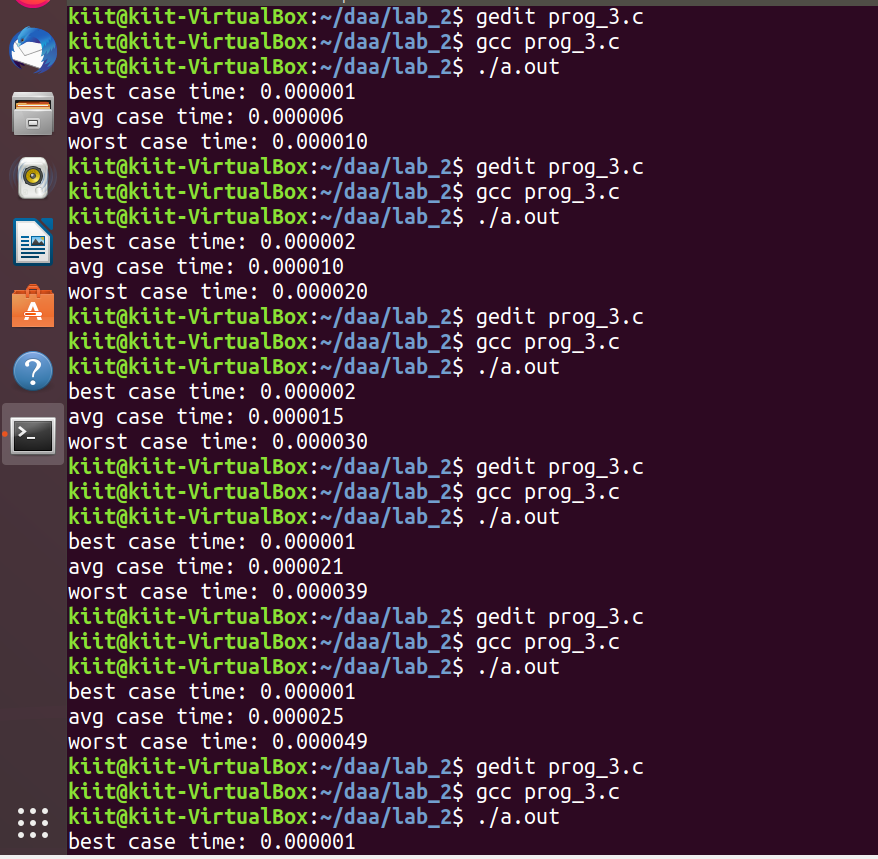
}

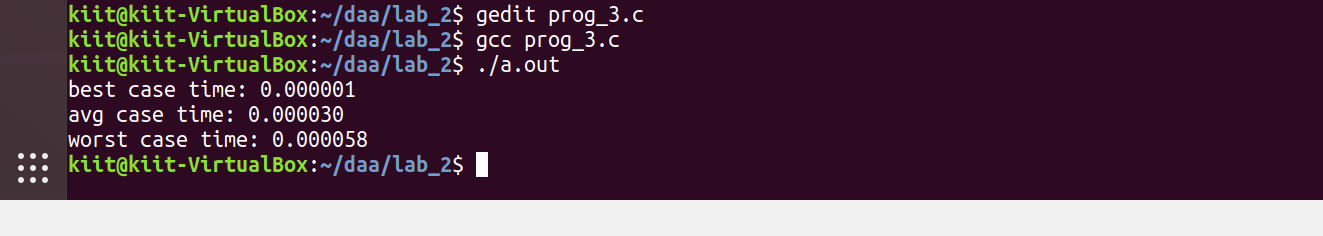
}

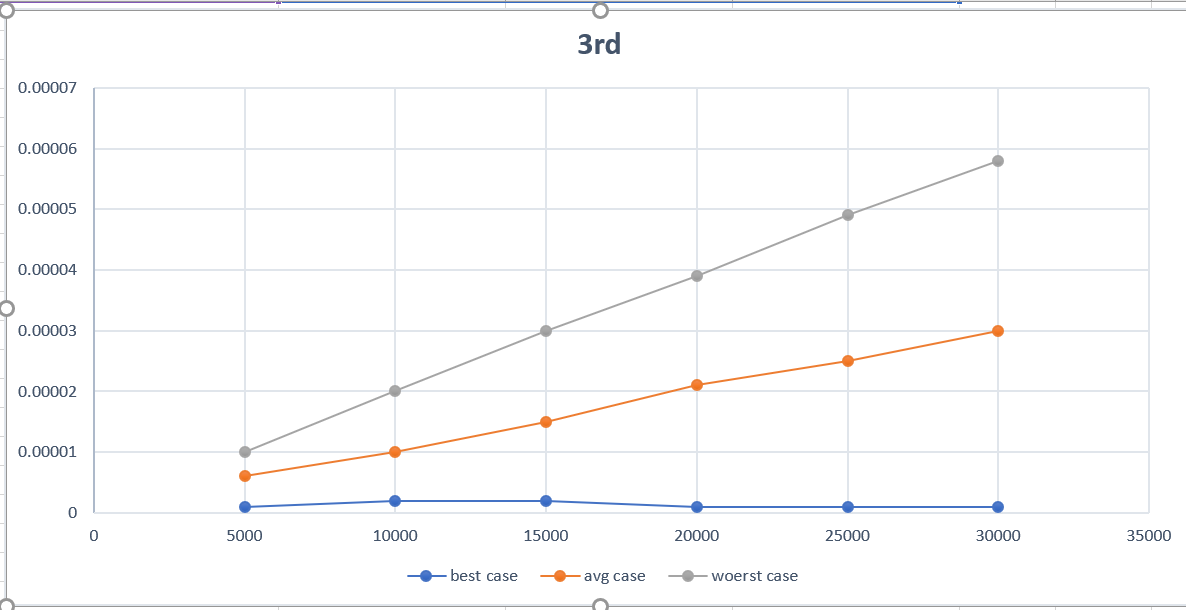
return 0;

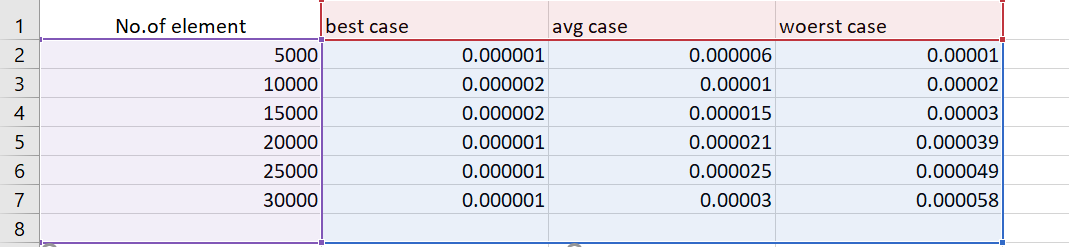
}

Output::









4.WAP in C to store 1 million integers in an array. To search an element in that array and find out its time complexity using binary search (best, worst, and average).

Program:

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int main() {

int n =30000;

int a[n];

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

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

a[i] = i + 1;

}

int best = a[(n - 1) / 2];

int worst = a[1];

int avg = a[n / 16];

clock\_t strt, end;

int low = 0, high = n - 1;

strt = clock();

while (low < high)

{

int mid = (low + high) / 2;

if (a[mid] == best) {

end = clock();

double t = end - strt;

printf("best case: %f\n", (t / CLOCKS\_PER\_SEC));

break;

}

if (a[mid] > best)

{

high = mid;

}

else

{

low = mid + 1;

}

}

low = 0, high = n - 1;

strt = clock();

while (low < high)

{

int mid = (low + high) / 2;

if (a[mid] == avg) {

end = clock();

double t = end - strt;

printf("avg case: %f\n", (t / CLOCKS\_PER\_SEC));

break;

}

if (a[mid] > avg)

{

high = mid;

}

else

{

low = mid + 1;

}

}

low = 0, high = n - 1;

strt = clock();

while (low < high)

{

int mid = (low + high) / 2;

if (a[mid] == worst) {

end = clock();

double t = end - strt;

printf("worst case: %f\n", (t / CLOCKS\_PER\_SEC));

break;

}

if (a[mid] > worst)

{

high = mid;

}

else

{

low = mid + 1;

}

}

return 0;

}

Output::

