**Student Name** : Pritam Kumar

**Student ID** : 11615236

**Email Address** : [thepritamkumar@gmail.com](mailto:thepritamkumar@gmail.com)

**GitHub Link** : <https://github.com/priitam/K1607-A02>

**Code :**

**1. Multithreading code 1**

#include<stdio.h>

#include<pthread.h>

//Global Declaration of varaiable and function.

void \*average\_calc();

void \*maxi();

void \*mini();

void \*display();

int x,y,a[100];

float avg1;

int n;

int main()

{

scanf("%d",&n);

printf("\n enter elements:\n ");

for (int i=0;i<n;i++) //taking input from user Complexity bigO(n)

{

printf("\nEnter %d element :",i);

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

}

pthread\_t th1, th2, th3, th4; // thread declaration

pthread\_create(&th1,NULL,average\_calc,NULL);

sleep(1);

pthread\_create(&th2,NULL,maxi,NULL);

sleep(1);

pthread\_create(&th3,NULL,mini,NULL);

sleep(1);

pthread\_create(&th4,NULL,display,NULL);

pthread\_join(th1,NULL);

pthread\_join(th2,NULL);

pthread\_join(th3,NULL);

pthread\_join(th4,NULL);

}

void \*average\_calc() //calculation of average value

{

float sum=0;

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

{

sum=sum+a[i];

}

avg1=sum/n;

}

void \*maxi() //calculation of maximum element

{

x=a[0];

for (int i=1;i<n;i++) // Complexity bigO(n)

{

if(a[i]>x)

x=a[i];

}

}

void \*mini() // calculation of minimum element

{

y=a[0];

for(int i=1;i<n;i++) // Complexity BigO(n)

{

if(a[i]<y)

y=a[i];

}

}

void \*display() //displaying all result

{

printf("\n The numbers are : ");

for (int i=0;i<n;i++) //Complexity bigO(n)

printf(" %d ",a[i]);

printf("\n The average of numbers : %0.2f",avg1);

printf("\n The maximum number is : %d",x);

printf("\n The minimum number is : %d",y);

printf("\n");

}

**2. Multithreading code 2**

#include<unistd.h>

#include<stdio.h>

#include<pthread.h>

// global variables and function declaration

int a[100],b[50],c[50];

int n;

int l,h;

int m;

void \*divide();

void \*h1();

void \*h2();

void \*join();

pthread\_mutex\_t l1; // lock creation

int main()

{

printf("Enter the number of element u want in array : ");

scanf("%d",&n);

m=n/2;

l=0;

h=n;

int x;

for(x=0;x<n;x++)

{

printf("\n Enter the %d element : ",x+1);

scanf("%d",&a[x]);

}

pthread\_mutex\_init(&l1,NULL);

pthread\_t th1;

printf("\n original array\n");

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

printf("%d , ",a[i]);

pthread\_create(&th1,NULL,divide,NULL);

pthread\_join(th1,NULL);

int i;

printf("\n-------------------------------------------------------------------------\n\t\t\tFINAL SORTED ARRAY IS :\n");

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

printf(" %d , ",a[i]);

printf("\n\n------------------------------------------------------------------------------------------------------\n\n");

}

void \*divide() // main thread

{

pthread\_t th2,th3, th4;

pthread\_create(&th2,NULL,h1,NULL);

sleep(1);

pthread\_create(&th3,NULL,h2,NULL);

sleep(1);

pthread\_create(&th4,NULL,join,NULL);

pthread\_join(th2,NULL);

pthread\_join(th3,NULL);

pthread\_join(th4,NULL);

printf("\n------------------------------------------------------------------------------\n\t\tMERGING SORT THREAD 1 AND SORT THREAD 2\n ");

}

void \*h1() // sorting thread 1

{

pthread\_mutex\_lock(&l1);

int i,j,t;

for(i=0;i<m;i++)

b[i]=a[i];

printf("\n------------------------------------------------------------------------------\n\t\t\tSORTING THREAD 1 :\n ");

for(i=0;i<m;i++) // Complexity bigO(n^2)

printf("%d , ",b[i]);

for (i=0;i<m-1;i++)

for(j=0;j<m-i-1;j++)

{

if(b[j]>b[j+1])

{

t=b[j];

b[j]=b[j+1];

b[j+1]=t;

}

pthread\_mutex\_unlock(&l1);

}

printf("\n\n\t\t\tSORTING DONE IN THREAD 1 : \n");

for(i=0;i<m;i++)

printf("%d , ",b[i]);

}

void \*h2() // sorting thread 2

{

pthread\_mutex\_lock(&l1);

int i,j,t;

for(i=0,j=m;j<n;j++,i++)

c[i]=a[j];

printf("\n------------------------------------------------------------------------------------\n\t\t\tSORTING THREAD 2 :\n");

for(i=0;i<n-m;i++) // Complexity bigO(n^2)

printf("%d , ",c[i]);

for (i=0;i<n-m-1;i++)

for(j=0;j<n-m-i-1;j++)

{

if(c[j]>c[j+1])

{

t=c[j];

c[j]=c[j+1];

c[j+1]=t;

}

}

printf("\n\n\t\t\tSORTING DONE IN THREAD 2 : \n");

for(i=0;i<n-m;i++)

printf("%d , ",c[i]);

pthread\_mutex\_unlock(&l1);

}

void \*join() // merging two sorted array

{

pthread\_mutex\_lock(&l1);

int i=0,j=0,k=0;

while(i<m && j<n-m) // Complexity bigO(n)

{

if(b[i]<c[j])

{

a[k]=b[i];

i++;

}

else

{

a[k]=c[j];

j++;

}

k++;

}

if(i>=n-m)

while(j<n-m)

{

a[k]=c[j];

j++;k++;

}

if(j>=m)

while(i<m)

{

a[k]=b[i];

i++;k++;

}

pthread\_mutex\_unlock(&l1);

}

In operating system the various processes are made multithreading so that they can handled efficiently by kernel thread. This increases the efficiency of program and reduces the burst time of a process. So as a programmer of operating system our main aim should be proper execution of all threads. Each process are modularized into various section and each section is allotted separate thread to increase the speed of execution.

Thread are of various types for various operating system.

POSIX thread for linux based system. WINDOW thread for windows and JAVA thread for java environment.

Thread\_create is used for creation for thread. It has four parameters first one to assign the name of thread. Second for setting up the attribute of thread. Third for calling the function which will be assigned to thread. Fourth and last one to pass the parameter to the thread.

Thread\_join is used to join the thread, it has two parameters. First is for name of the thread nad second on for returning value if any.

COMPLEXITY:

In assigned problem 1 loops has been used to traverse the element in array and also for doing comparison.

Complexity is Big’Oh(n)

In assigned problem 2 loops has been used for taking input in array and also bubble sort has been used to sort the element.

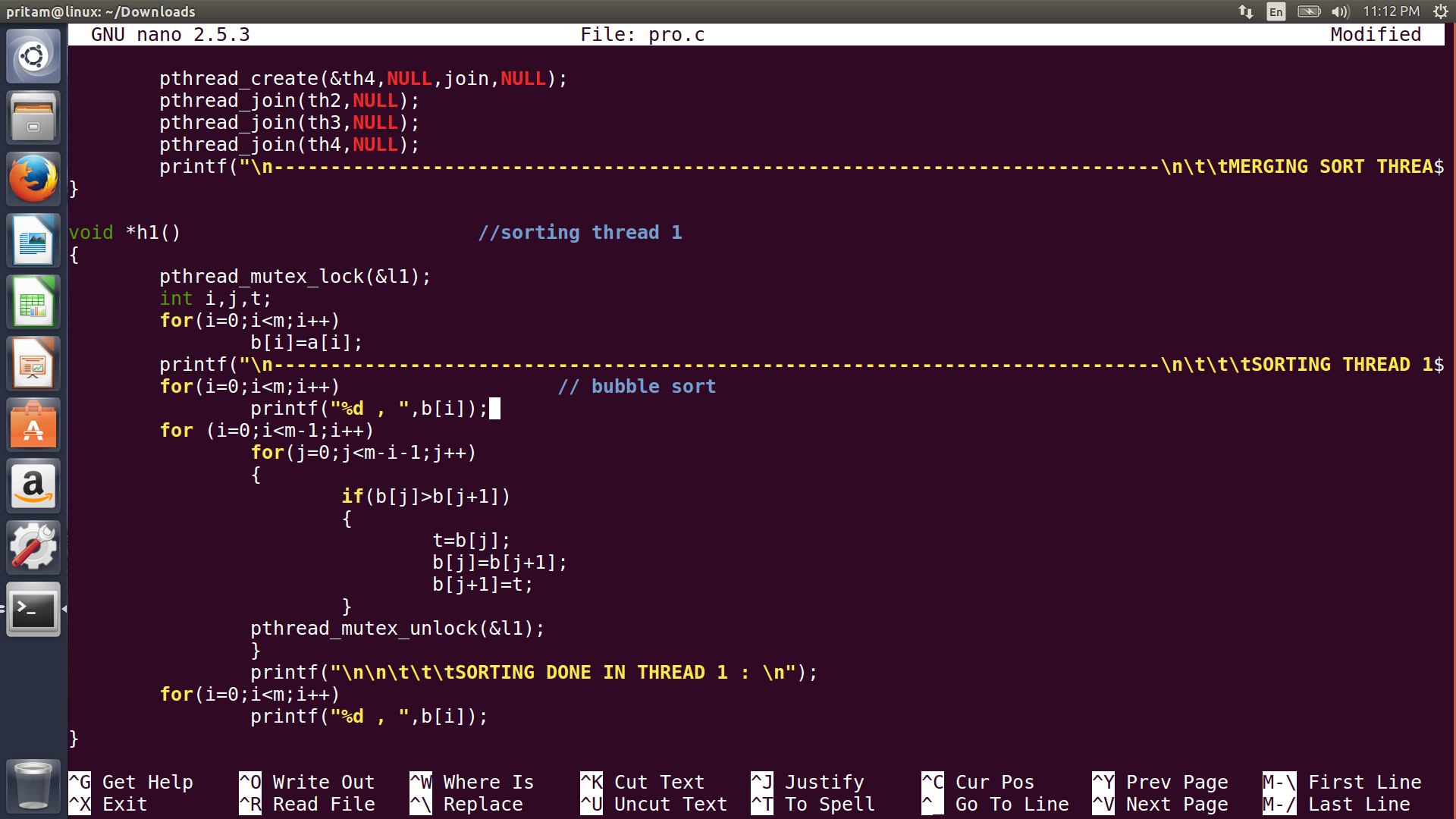
Complexity is Big’Oh(n^2).

BOUNDARY CONDITION AND CONSTRAINTS:

The code fails in time efficiency and space complexities when we have large number of element to process.

It can easily process data upto 100 elements.

Memory has been allocated statically just to show the demo processing of the programs it can be implemented through dynamic programming concept by asking any number of element from user.



TEST CASES:

Code has been successfully compiled and run for various test cases and it is found accurate in most of cases.

