Problem: To find the area of the given geometrical figure through maxium CPU utilization.

Approach:We divide problem into independent sub-problems and solve them simultaneously inindependent threads.

Language: UNIX 'C' (POSIX thread library)

Method1:Allowing the main thread to wait for other thread to finish their jobs and thereby summing up the total area computer by the threads.

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

#include <math.h>

#define NUM\_THREADS 5

struct Triangle{

float base,height,area;

};

struct SemiCircle{

float radius,area;

};

struct Rectangle{

float breadth,length,area;

};

struct Square{

float side,area;

};

struct Trapezium{

float p1,p2,height,area;

};

void \*AreaOfSemiCircle(void \*dimension){

struct SemiCircle\* temp = (struct SemiCircle \*)dimension;

temp->area=3.141592654\*temp->radius\*temp->radius/2;

printf("AreaOfSemiCircle: %f\n",temp->area);

}

void \*AreaOfTriangle(void \* dimension){

struct Triangle\* temp=(struct Triangle\*)dimension;

temp->area=(temp->base\*temp->height)/2;

printf("AreaOfTriangle: %f\n",temp->area);

}

void \*AreaOfTrapezium(void \* dimension){

struct Trapezium\* temp=(struct Trapezium\*)dimension;

float sum=( temp->p1) +( temp->p2 );

printf("\nsum= %f and p1=%f,p2=%f,height=%f,sum\* height= %f\n",sum,temp->p1,temp->p2,temp->height,sum\*temp->height);

temp->area= sum\* (temp->height/2);

printf("AreaOfTrapezium: %f\n",temp->area);

}

void main (void){

pthread\_t th[6];

struct Triangle t1,t2;

t1.base=50; t1.height=20;

t2.base=100; t2.height=20;

struct SemiCircle sc1;

sc1.radius=45/2;

struct Trapezium tz1,tz2,tz3;

tz1.p1=50; tz1.p2=80; tz1.height=20;

tz2.p1=80; tz2.p2=65; tz2.height=45;

tz3.p1=65; tz3.p2=100; tz3.height=30;

pthread\_create(&th[0],NULL,AreaOfTriangle,(void\*)&t1);

pthread\_create(&th[1],NULL,AreaOfTriangle,(void\*)&t2);

pthread\_create(&th[2],NULL,AreaOfSemiCircle,(void\*)&sc1);

pthread\_create(&th[3],NULL,AreaOfTrapezium,(void\*)&tz1);

pthread\_create(&th[4],NULL,AreaOfTrapezium,(void\*)&tz2);

pthread\_create(&th[5],NULL,AreaOfTrapezium,(void\*)&tz3);

pthread\_join(th[0],NULL);

pthread\_join(th[1],NULL);

pthread\_join(th[2],NULL);

pthread\_join(th[3],NULL);

pthread\_join(th[4],NULL);

pthread\_join(th[5],NULL);

float totalArea=t1.area+t2.area+sc1.area+tz1.area+tz2.area+tz3.area;

printf("total area of given geometrical figure : %f\n",totalArea);

}

Method2:We define a global variable totalArea which is shared by all the threads and each thread after computing the area adds their result to this global variable and hence main thread doesn't need to wait for rest of the threads. But this method faces the problem of data inconsistency due to concurrent access of the shared global variable totalArea, therefore we need to provide synchronization between threads while accessing shared variable. This is achieved by using pthread\_mutex\_lock function.

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include<sys/sem.h>

#include<sys/types.h>

#include<sys/ipc.h>

#define PIE 3.141592654

struct Triangle{

float base,height;

};

struct SemiCircle{

float radius;

};

struct Rectangle{

float breadth,length;

};

struct Square{

float side;

};

struct Trapezium{

float p1,p2,height;

};

float totalArea=0;

pthread\_mutex\_t mutex=PTHREAD\_MUTEX\_INITIALIZER;

void \*AreaOfSemiCircle(void \*dimension){

float area;

int rc;

struct SemiCircle temp = \*((struct SemiCircle \*) dimension);

area=PIE\*temp.radius\*temp.radius/2;

pthread\_mutex\_lock(&mutex);

/\*\*critical section\*/

totalArea+=area;

/\*\*critical section\*/

pthread\_mutex\_unlock(&mutex);

printf("AreaOfSemiCircle: %f and totalArea: %f\n",area,totalArea);

}

void \*AreaOfTriangle(void \* dimension){

struct Triangle temp=\*(struct Triangle\*)dimension;

float area=(temp.base\*temp.height)/2;

pthread\_mutex\_lock(&mutex);

/\*\*critical section\*/

totalArea+=area;

/\*\*critical section\*/

pthread\_mutex\_unlock(&mutex);

printf("AreaOfTriangle: %f and totalArea: %f\n",area,totalArea);

}

void \*AreaOfTrapezium(void \* dimension){

struct Trapezium temp=\*(struct Trapezium\*)dimension;

float area=((temp.p1+temp.p2)\*temp.height)/2;

pthread\_mutex\_lock(&mutex);

/\*\*critical section\*/

totalArea+=area;

/\*\*critical section\*/

pthread\_mutex\_unlock(&mutex);

printf("AreaOfTrapezium: %f and totalArea: %f\n",area,totalArea);

}

void main (void){

pthread\_t th[6];

struct Triangle t1,t2;

t1.base=50; t1.height=20;

t2.base=100; t2.height=20;

struct SemiCircle sc1;

sc1.radius=45/2;

struct Trapezium tz1,tz2,tz3;

tz1.p1=50; tz1.p2=80; tz1.height=20;

tz2.p1=80; tz2.p2=65; tz2.height=45;

tz3.p1=65; tz3.p2=100; tz3.height=30;

pthread\_create(&th[0],NULL,AreaOfTriangle,(void\*)&t1);

pthread\_create(&th[1],NULL,AreaOfTriangle,(void\*)&t2);

pthread\_create(&th[2],NULL,AreaOfSemiCircle,(void\*)&sc1);

pthread\_create(&th[3],NULL,AreaOfTrapezium,(void\*)&tz1);

pthread\_create(&th[4],NULL,AreaOfTrapezium,(void\*)&tz2);

pthread\_create(&th[5],NULL,AreaOfTrapezium,(void\*)&tz3);

}