**CPP LAB**

**ASSIGNMENT-1**

**NAME-**ABHISHEK TIBREWAL

**ID-**2016UCP1103

**BATCH-**A (1, 2)

1. **Write a program to create five threads in C.**

**CODE:**

#include<stdio.h>

#include<pthread.h>

void \*fun(void \*a) //the argument must be of type void\*

{

char \*b;

b=(char \*)a;

printf("%s\n",b);

sleep(10); //provide gap

return(NULL);

}

int main()

{

pthread\_t t1,t2,t3,t4,t5;

char \*m1="thread 1:";

char \*m2="thread 2:";

char \*m3="thread 3:";

char \*m4="thread 4:";

char \*m5="thread 5:";

pthread\_create(&t1,NULL,fun,(void \*)m1); //cast argumnt to void\*

printf(" THREAD1 CREATED::YOU ARE IN MAIN FUNCTION\n");

pthread\_create(&t2,NULL,fun,(void \*)m2);

pthread\_create(&t3,NULL,fun,(void \*)m3); //cast argumnt to void\*

printf(" THREAD3 CREATED::YOU ARE IN MAIN FUNCTION\n");

pthread\_create(&t4,NULL,fun,(void \*)m4);

pthread\_create(&t5,NULL,fun,(void \*)m5); //cast argumnt to void\*

printf(" ALL THREADS CREATED::YOU ARE IN MAIN FUNCTION\n");

pthread\_join(t1,NULL);

pthread\_join(t2,NULL);

pthread\_join(t3,NULL);

pthread\_join(t4,NULL);

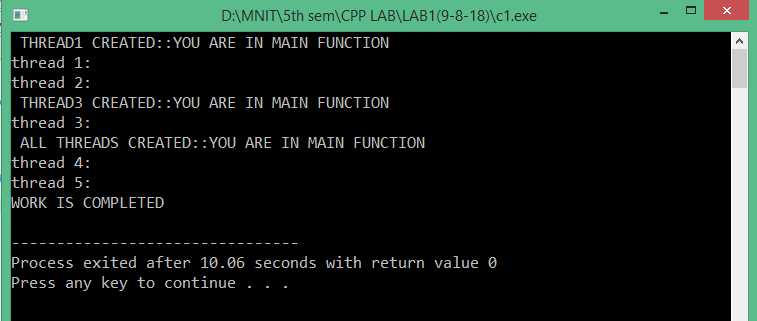
pthread\_join(t5,NULL);

printf("WORK IS COMPLETED\n");

return 0;

}

**OUTPUT:**

****

1. **Program to print “Hello World” using thread in C**

**CODE:**

#include<stdio.h>

#include<pthread.h>

void \* fun(void \*a) //function to be run when thread is created.

{

char \*b;

b=(char \*)a;

printf("%s",b);

}

int main()

{

char \*msg="hello world\n";

pthread\_t t1; //create a thread id

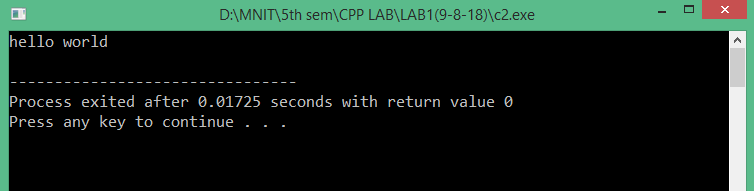
pthread\_create(&t1,NULL,fun,msg); //create thread

pthread\_join(t1,NULL); //for holding the main fun. so that before execution of thred,the main fun. can not be exit.

return 0;

}

**OUTPUT:**

****

1. **Program that computes the square roots of the integers from 0 to 99 in a separate thread and returns an array of doubles containing the \* results. In the meantime the main thread should display a short message to the user and then display the results of the computation**

**CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<math.h>

double \*a;

void \* fun(void \*x) //function to be run when thread is created.

{

int i;

a=(double \*)malloc(sizeof(double)\*100);

for(i=0;i<100;i++)

{

a[i]=(double)pow(i,0.5);

}

return((void \*)a);

}

int main()

{

void \* b;

pthread\_t t1; //create a thread id

pthread\_create(&t1,NULL,fun,NULL); //create thread

pthread\_join(t1,&b); //for holding the main fun. so that before execution of thred,the main fun. can not be exit.

int i;

a=(double \*)b; //returned value by function will store in second argument of pthread\_join.

for(i=0;i<100;i++)

{

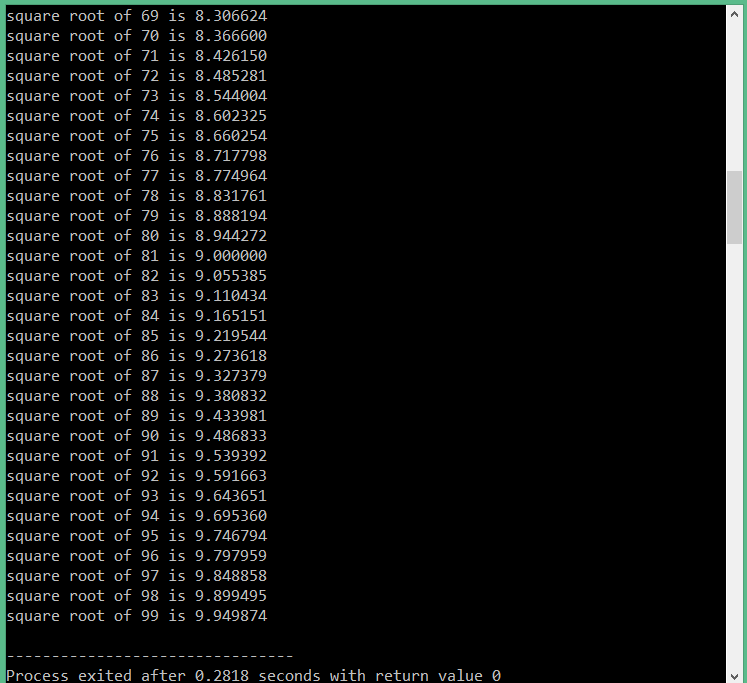
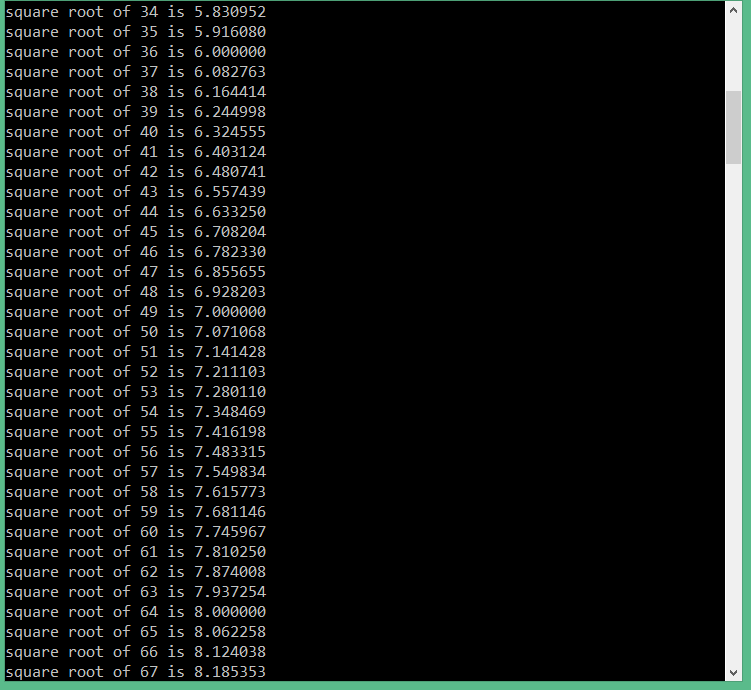
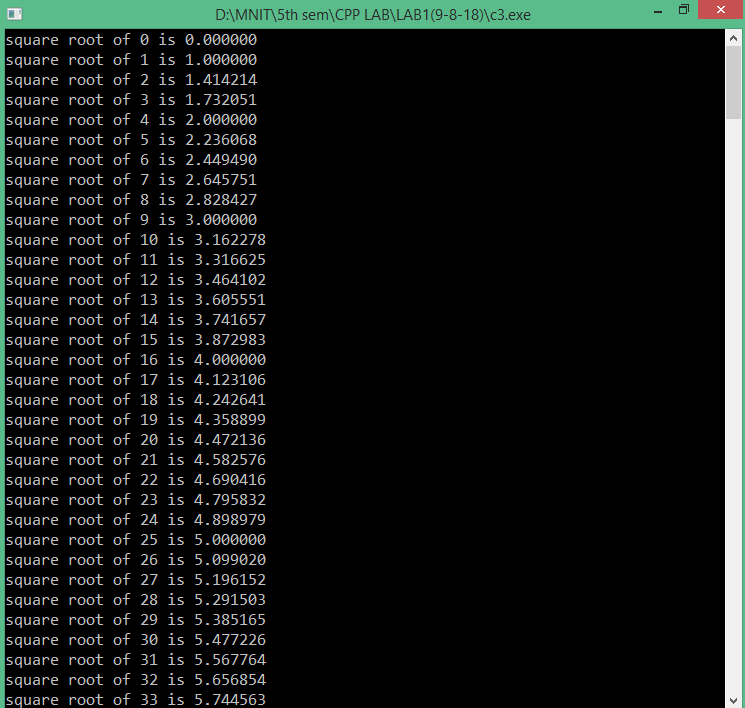
printf("square root of %d is %lf\n",i,a[i]);

}

return 0;

}

**OUTPUT:**

****

1. **Implement a linked list as a Parallel program (based on Pthreads) with Mutex locks for the entire linked list Implementation should support Search( ), Insert( ), and Delete( ) functions.**

**CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<math.h>

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

int ch;

int count=0;

struct node

{

int info;

struct node \*link;

};

struct node\* START=NULL;

struct node\* create\_node()

{

struct node \*n;

n=(struct node \*)malloc(sizeof(struct node));

return n;

}

void insert\_node()

{

struct node \*temp,\*t;

temp=create\_node();

count++;///important used in insert mid and delete

printf("enter an element to insert");

scanf("%d",&temp->info);

temp->link=NULL;

if (NULL==START)

{

START=temp;

}

else

{

t=START;

while(t->link!=NULL)

{

t=t->link;

}

t->link=temp;

}

printf("%d is inserted\n",temp->info);

}

void first\_delet()

{

struct node \*r;

if(NULL==START)

{

printf("list is empty");

}

else

{

r=START;

START=START->link;

count=count-1;

printf("deleted element is:%d\n",r->info);

free(r);

}

}

int search(int value)

{

struct node\* curr = START;

while(curr != NULL)

{

if(curr->info==value)

return 1;

curr= curr->link;

}

return 0;

}

void \* fun(void \*x) //function to be run when thread is created.

{

if(ch==1)

{

pthread\_mutex\_lock(&flag);

insert\_node();

pthread\_mutex\_unlock(&flag);

}

if(ch==2)

{

pthread\_mutex\_lock(&flag);

first\_delet();

pthread\_mutex\_unlock(&flag);

}

if(ch==3)

{

pthread\_mutex\_lock(&flag);

int x;

printf("enter the value to be search\n");

scanf("%d",&x);

if(search(x)==1)

{

printf("%d is present in list\n",x);

}

else

{

printf("%d is not present in list\n",x);

}

pthread\_mutex\_unlock(&flag);

}

return NULL;

}

int main()

{

pthread\_t t;

while(1)

{

printf("1 for insert\n2 for delete\n3 for search\n4 for exit\n");

scanf("%d",&ch);

if(ch<=3)

{

pthread\_create(&t,NULL,&fun,NULL);

}

sleep(5);

if(ch==4)

{

break;

}

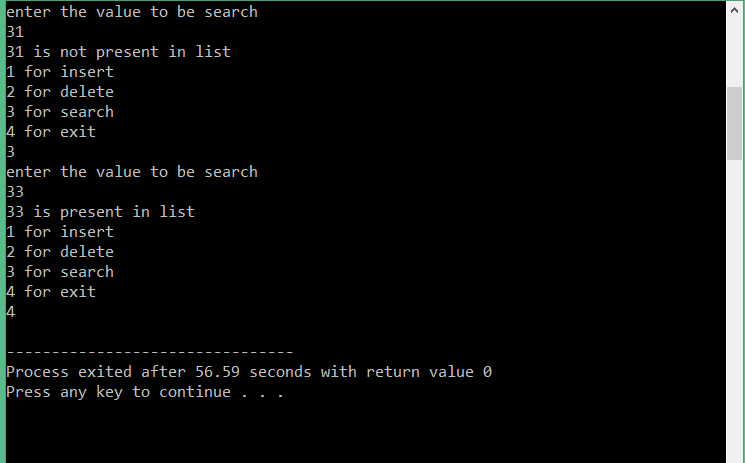
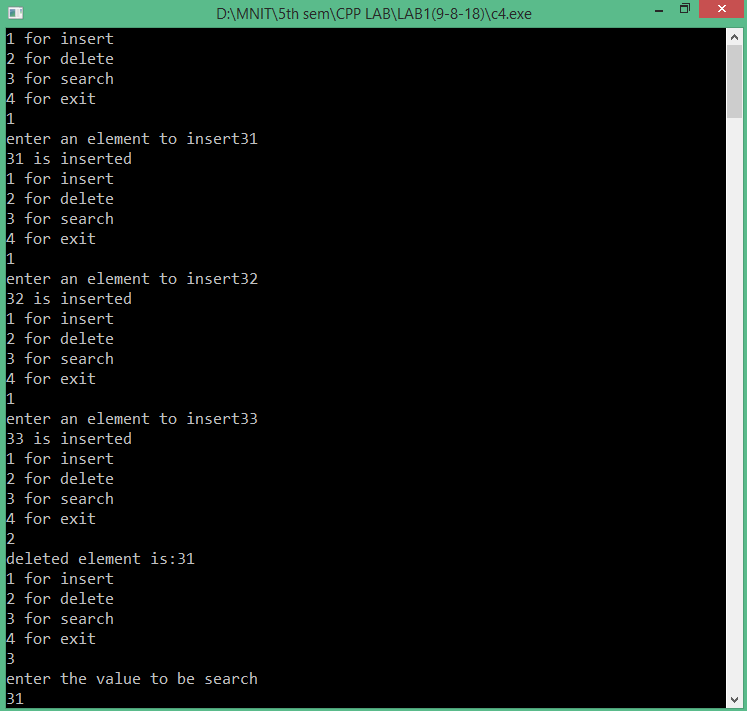
}

pthread\_join(t,NULL);

return 0;

}

**OUTPUT:**

****

1. **Program to computes the total of the values of the matrix using thread and mutex lock on global variable total**

**CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

#include<math.h>

int inp[10][10];

int r,c,s=0,f;

int co=-1;

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

void \* fun(void \*x) //function to be run when thread is created.

{

pthread\_mutex\_lock(&flag);

int i;

co++;

f=0;

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

{

f+=inp[co][i];

}

sleep(3);

s=s+f;

printf("%d\n",f);

pthread\_mutex\_unlock(&flag);

return NULL;

}

int main()

{

int x,y;

printf("enter no. of rows and columns respectively\n");

scanf("%d%d",&r,&c);

printf("enter the array elements\n");

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

{

for(y=0;y<c;y++)

{

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

}

}

pthread\_t t[r];

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

{

pthread\_create(&t[x],NULL,&fun,NULL);

}

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

{

pthread\_join(t[x],NULL);

}

printf("the sum is::%d",s);

return 0;

}

**OUTPUT:**

