**CPP Lab**

**Assignment-1**

**Name-Ashish Goyal**

**Id-2016ucp1100**

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

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

**CODE-**

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

void\* func(void \*myvar)//function related to thread

{

char\* msg;

msg=(char \*) myvar;//as receive void, so type cast to message type

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

sleep(2);

}

int main(int argc,char\* argv[])

{

pthread\_t thread1,thread2,thread3,thread4,thread5;//to create a thread\_id

char \*msg1="first Thread";

char \*msg2="second Thread";

char \*msg3="third Thread";

char \*msg4="fourth Thread";

char \*msg5="fifth Thread";

pthread\_create(&thread1,NULL,&func,(void \*)msg1);//it create a thread and starts its execution simultaneously with main func

pthread\_create(&thread2,NULL,&func,msg2);

printf("after creating thread1 and 2\n");

pthread\_create(&thread3,NULL,&func,msg3);

pthread\_create(&thread4,NULL,&func,msg4);

pthread\_create(&thread5,NULL,&func,msg5);

printf("all 5 threads are created\n");

pthread\_join(thread1,NULL);//join thread with main so that it completes its execution before main terminates

pthread\_join(thread2,NULL);

pthread\_join(thread3,NULL);

pthread\_join(thread4,NULL);

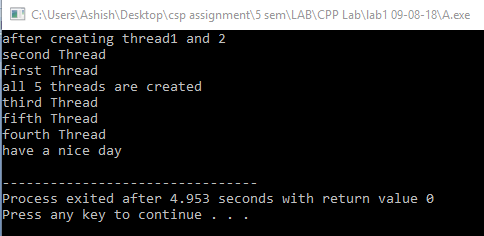
pthread\_join(thread5,NULL);

printf("have a nice day\n");

return 0;

}

**OUTPUT**



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

**CODE-**

#include<stdio.h>

#include<stdlib.h>

#include<pthread.h>

void\* func(void \*myvar)

{

printf("Hello World............\n");

sleep(1);

}

int main(int argc,char\* argv[])

{

pthread\_t thread1;

pthread\_create(&thread1,NULL,&func,NULL);

printf("thread is created\n");

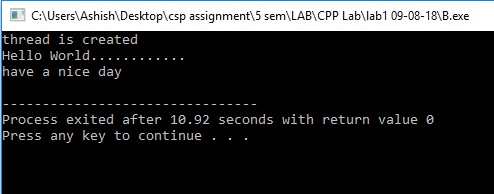
pthread\_join(thread1,NULL);

printf("have a nice day\n");

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 \*arr;

void\* func(void \*myvar)

{

int i;

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

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

{

double temp;

temp=(double)sqrt(i);

arr[i]=temp;

}

return ((void \*)arr);

}

int main(int argc,char\* argv[])

{

void \* temp;//for holding returned result

// void \* temp;//for holding returned result

int i;

pthread\_t thread1;

pthread\_create(&thread1,NULL,&func,NULL);

pthread\_join(thread1,&temp); //here it is returned

arr=(double \*)temp;

//pthread\_join(thread1,NULL); //here it is returned

//arr=(double \*)temp;

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

{

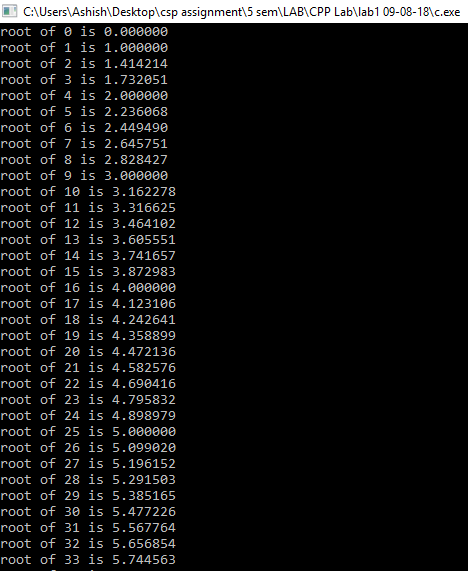
printf("root of %d is %lf\n",i,arr[i]);

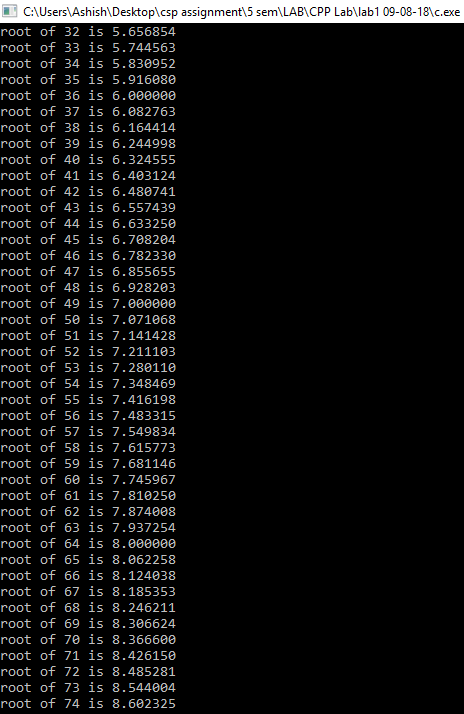
}

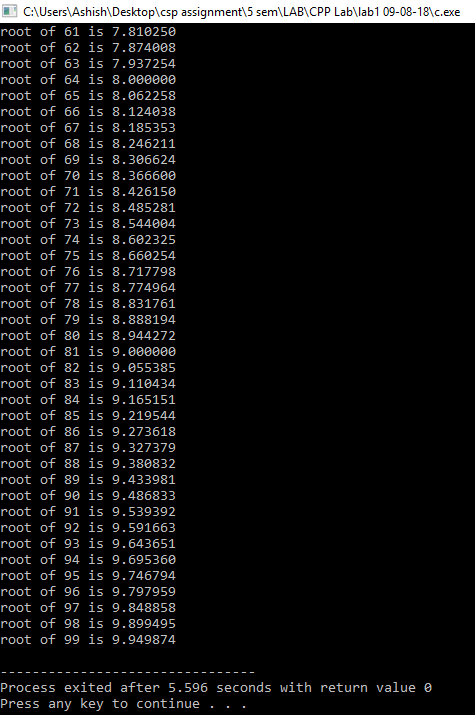
return 0;

}

**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>

int count=0,ch=0;

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

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();

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\t2 for delete\t3 for search\t4 for exit\n");

scanf("%d",&ch);

if(ch<=3)

{

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

}

sleep(5);

if(ch==4)

{

break;

}

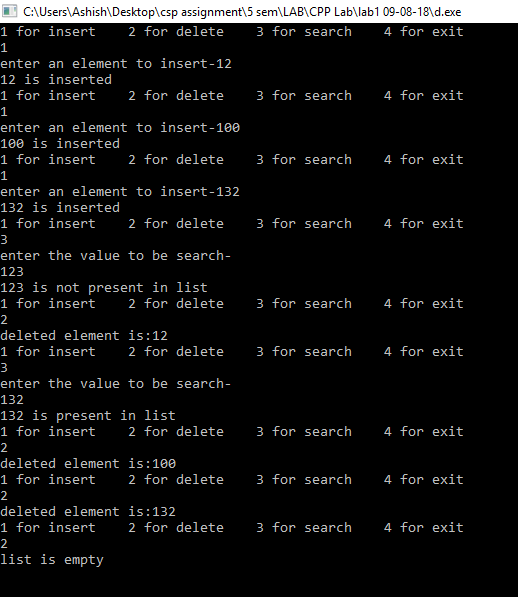
}

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 arr[10][10];

int l,m,sum=0,total;

int counter=-1;

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

//pthread\_mutex\_init(&lock,NULL);

void\* func(void\* arg)

{

pthread\_mutex\_lock(&lock);

int i;

counter++;

total=0;

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

{

total+=arr[counter][i];

}

sleep(1);

sum+=total;

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

// sleep(1);

pthread\_mutex\_unlock(&lock);

return NULL;

}

int main(int argc,char\* argv[])

{

int i,j;

printf("enter row and column\n");

scanf("%d %d",&l,&m);

printf("fill array\n");

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

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

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

pthread\_t thread[l];

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

{

pthread\_create(&thread[i],NULL,&func,NULL);

}

// printf("%d\n",sum);

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

pthread\_join(thread[i],NULL);

printf("total sum is=%d\n",sum);

return 0;

}

**OUTPUT**

