**Name-Ujjawal Mandhani**

**Batch-F6**

**Enroll No.-9918103237**

**OSSP Lab – 5**

**Question 1:**Write simple thread program which creates 5 threadswith the pthread\_create() routine.Each thread prints a “Hello World!” message, and thenterminates with a call topthread\_exit( ).

**#include <iostream>**

**#include <cstdlib>**

**#include <pthread.h>**

**using namespace std;**

**#define NUM\_THREADS 5**

**void \*PrintHello(void \*threadid)**

**{**

**long tid;**

**tid = (long)threadid;**

**cout << "Hello World! Thread ID, " << tid << endl;**

**pthread\_exit(NULL);**

**}**

**int main()**

**{**

**pthread\_t threads[NUM\_THREADS];**

**int rc;**

**long i;**

**for (i = 0; i < NUM\_THREADS; i++)**

**{**

**cout << "main() : creating thread, " << i << endl;**

**rc = pthread\_create(&threads[i], NULL, PrintHello, (void \*)i);**

**if (rc)**

**{**

**cout << "Error:unable to create thread," << rc << endl;**

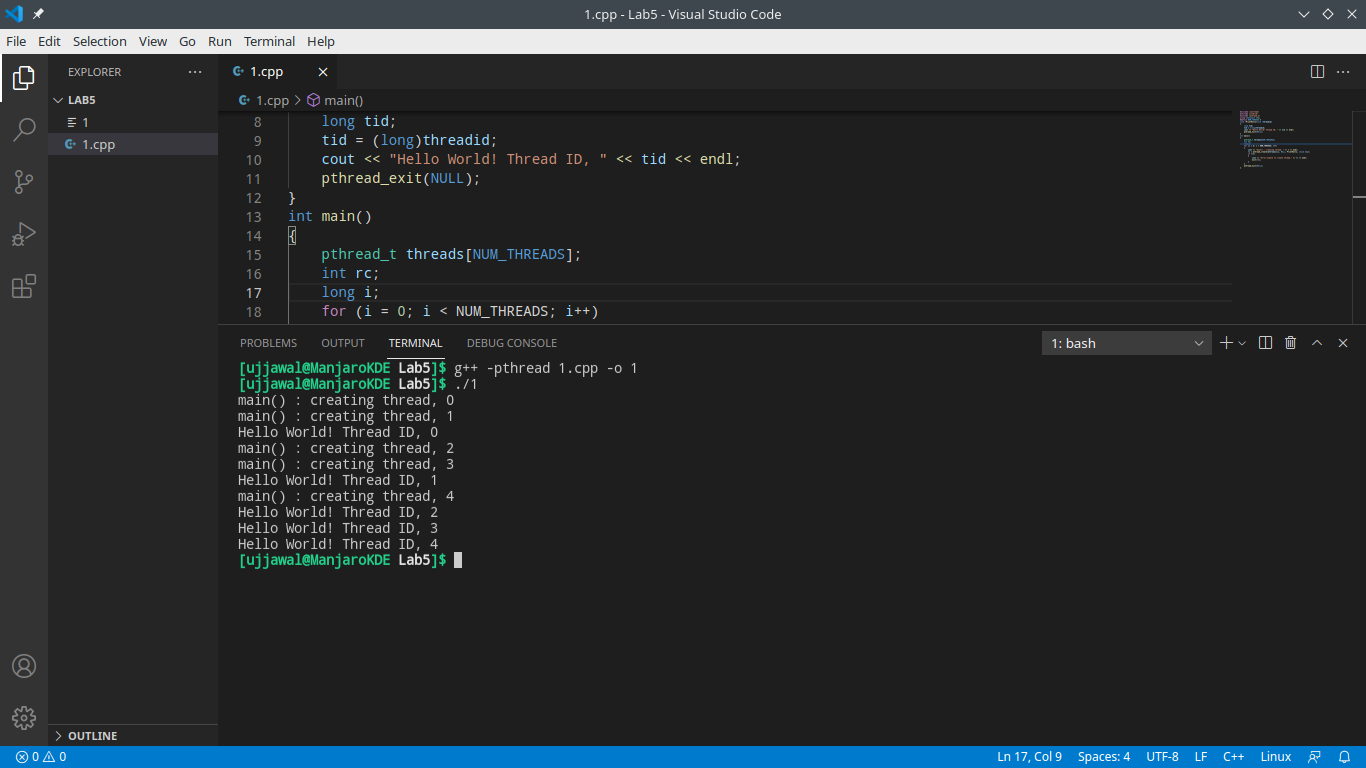
**exit(-1);**

**}**

**}**

**pthread\_exit(NULL);**

**}**

****

**Question 2:**Write a thread program which demonstrates how to “wait” for thread completions by usingthe Pthread join routine. Since some implementationsof Pthreads may not create threads in ajoinable state, therefore explicitly created attributein a joinable state so that they can bejoined later. Created thread should perform the calculationof sum =sum + sin(i) + Tan(i), wherei =0 to 1000.

**#include <pthread.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

**#define NUM\_THREADS 4**

**void \*BusyWork(void \*t)**

**{**

**int i;**

**long tid;**

**double result = 0.0;**

**tid = (long)t;**

**printf("Thread %ld starting...\n", tid);**

**for (i = 0; i < 1000; i++)**

**{**

**result = result + sin(i) + tan(i);**

**}**

**printf("Thread %ld done. Result = %e\n", tid, result);**

**pthread\_exit((void \*)t);**

**}**

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

**{**

**pthread\_t thread[NUM\_THREADS];**

**pthread\_attr\_t attr;**

**int rc;**

**long t;**

**void \*status;**

**pthread\_attr\_init(&attr);**

**pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_JOINABLE);**

**for (t = 0; t < NUM\_THREADS; t++)**

**{**

**printf("Main: creating thread %ld\n", t);**

**rc = pthread\_create(&thread[t], &attr, BusyWork, (void \*)t);**

**if (rc)**

**{**

**printf("ERROR; return code from pthread\_create()is %d\n", rc);**

**exit(-1);**

**}**

**}**

**pthread\_attr\_destroy(&attr);**

**for (t = 0; t < NUM\_THREADS; t++)**

**{**

**rc = pthread\_join(thread[t], &status);**

**if (rc)**

**{**

**printf("ERROR; return code from pthread\_join() is%d\n", rc);**

**exit(-1);**

**}**

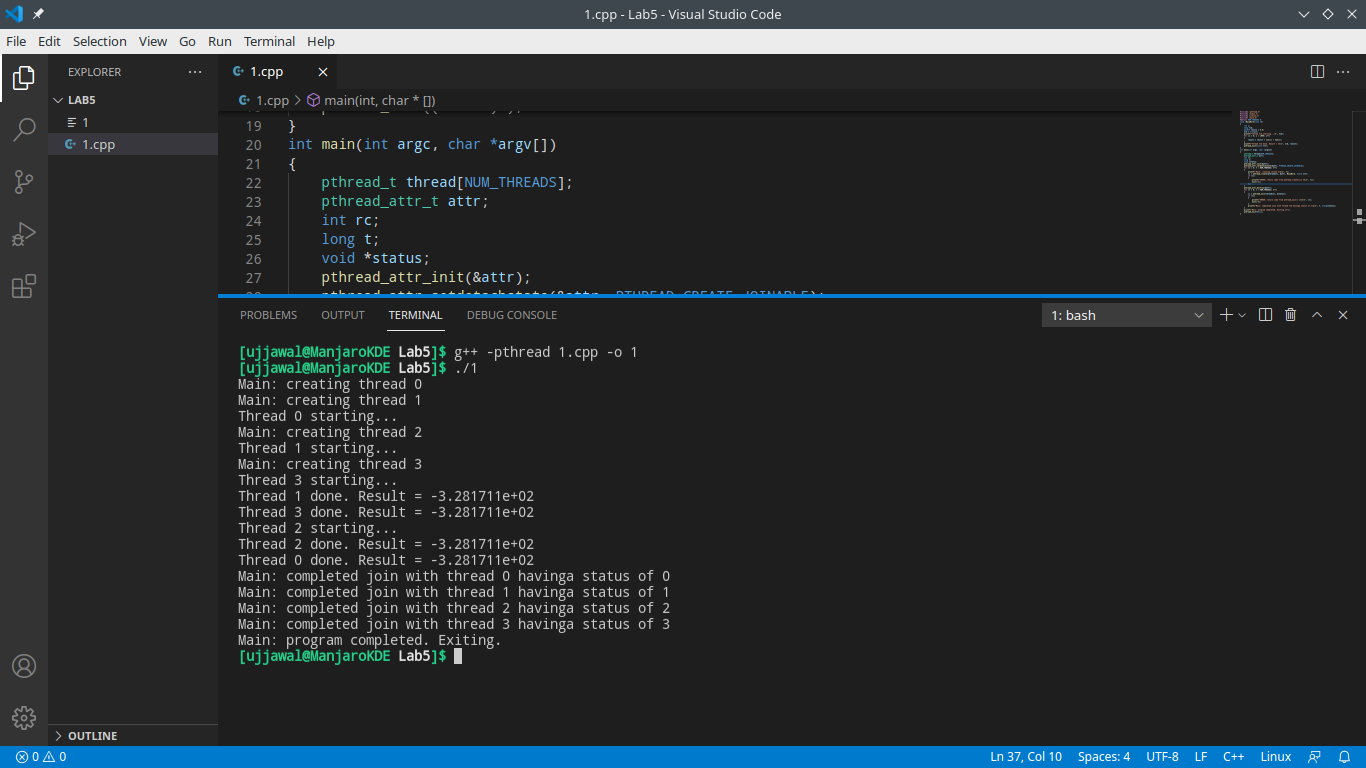
**printf("Main: completed join with thread %ld havinga status of %ld\n", t, (long)status);**

**}**

**printf("Main: program completed. Exiting.\n");**

**pthread\_exit(NULL);**

**}**



**Question 3:**pthread\_exit(NULL);}3. Write a program that computes the square rootsof the integers from 0 to 10 in aseparate thread and returns an array of doubles containingthe results. In the meantimethe main thread should display a short message tothe user and then display the resultsof the computation when they are ready

**#include <pthread.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

**void \*thread\_function(void \*param)**

**{**

**int i;**

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

**// createarray to store results.**

**for (i = 0; i < 11; i++)**

**{**

**result[i] = sqrt((double)i);**

**// compute the answerusing sqrt() library function.**

**}**

**return (void \*)result;**

**// return the result.**

**}**

**int main()**

**{**

**pthread\_t threadID;**

**void \*exit\_status;**

**double \*result;**

**int i;**

**pthread\_create(&threadID, NULL, thread\_function, NULL);**

**pthread\_join(threadID, &exit\_status);**

**printf("\n\nResults are being computed. Please wait.\n\n");**

**result = (double \*)exit\_status;**

**for (i = 0; i < 11; i++)**

**{**

**printf("\nsqrt(%d) = %lf", i, result[i]);**

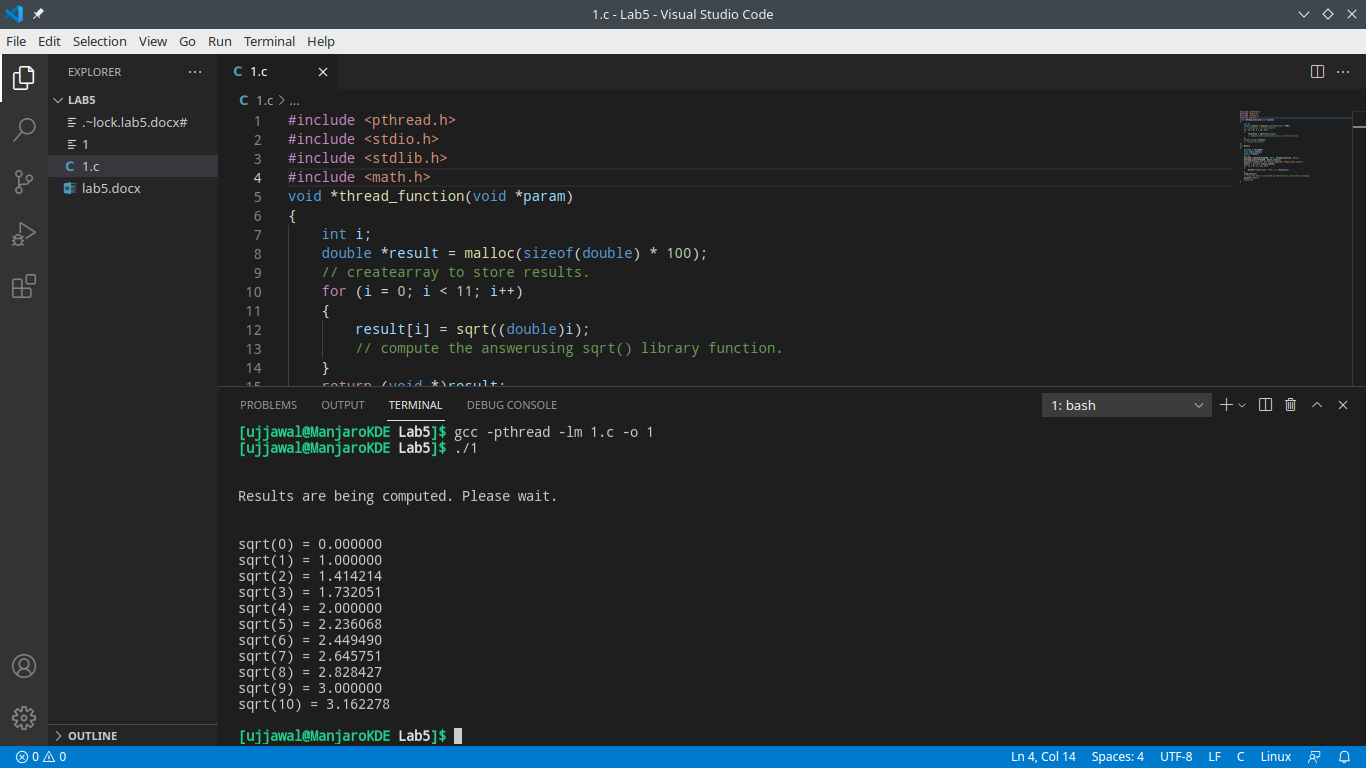
**}**

**free(result);**

**// free the memory allocated by thethread to avoid memory leakage.**

**printf("\n\n");**

**return 0;**

**}**

**Question 4:**Demonstration of Race Condition in producer andconsumer problem using threadImplementation.

**void \*producer (void \*q)**

**{**

**fifo = (queue \*)q;**

**while (1)**

**{**

**do\_work(PRODUCER\_CPU, PRODUCER\_BLOCK);**

**while (fifo->full && \*total\_produced != WORK\_MAX)**

**{**

**printf ("prod %d:\t FULL.\n", my\_tid);**

**if (\*total\_produced == WORK\_MAX)**

**{**

**break;**

**}**

**item\_produced = (\*total\_produced)++;**

**queueAdd (fifo, item\_produced);**

**printf("prod %d:\t %d.\n", my\_tid, item\_produced);**

**}**

**return(NULL);**

**}**

**void \*consumer (void \*q)**

**{**

**fifo = (queue \*)q;**

**while (1)**

**{**

**while (fifo->empty && \*total\_consumed != WORK\_MAX)**

**{**

**printf ("con %d:\t EMPTY.\n", my\_tid);**

**}**

**if (\*total\_consumed == WORK\_MAX)**

**{**

**break;**

**}**

**queueRemove (fifo, &item\_consumed);**

**(\*total\_consumed)++;**

**do\_work(CONSUMER\_CPU,CONSUMER\_CPU);**

**printf ("con %d:\t %d.\n", my\_tid, item\_consumed);**

**}**

**return(NULL);**

**}**

One of the problems with this solution is that itcontains a race condition. Suppose the schedulercreated the following interleaving:

Producer Consumer

Remove an item from the queue.A check to see if the queue is empty showsthat it is.

Set queue->empty =1.

Add last items to the queue.Set queue->empty = 0i == WORK\_MAX, producer exits

Set queue->empty = 1.

Set queue->full = 0Start spinning

**Question 5:** Write a program to implement producer consumerproblem (Using MUTEX semaphores)

**#include <stdio.h>**

**#include <stdlib.h>**

**int temp = 0, overflow = 0, mutex = 1, underflow = 10;**

**int wait\_protocol(int counter)**

**{**

**return (--counter);**

**}**

**int signal\_protocol(int counter)**

**{**

**return (++counter);**

**}**

**void producer()**

**{**

**mutex = wait\_protocol(mutex);**

**overflow = signal\_protocol(overflow);**

**underflow = wait\_protocol(underflow);**

**temp++;**

**printf("\nProducer:\t Element %d\n", temp);**

**mutex = signal\_protocol(mutex);**

**}**

**void consumer()**

**{**

**mutex = wait\_protocol(mutex);**

**overflow = wait\_protocol(overflow);**

**underflow = signal\_protocol(underflow);**

**printf("\nConsumer:\t Element %d\n", temp);**

**temp--;**

**mutex = signal\_protocol(mutex);**

**}**

**int main()**

**{**

**int choice;**

**printf("\n1.Producer\t");**

**printf("2. Consumer\t");**

**printf("3. Quit\t");**

**while (1)**

**{**

**printf("\nEnter your choice:\t");**

**scanf("%d", &choice);**

**switch (choice)**

**{**

**case 1:**

**if ((mutex == 1) && (underflow != 0))**

**{**

**producer();**

**}**

**else**

**{**

**printf("Buffer Overflow\n");**

**}**

**break;**

**case 2:**

**if ((mutex == 1) && (overflow != 0))**

**{**

**consumer();**

**}**

**else**

**{**

**printf("Buffer Underflow\n");**

**}**

**break;**

**case 3:**

**exit(0);**

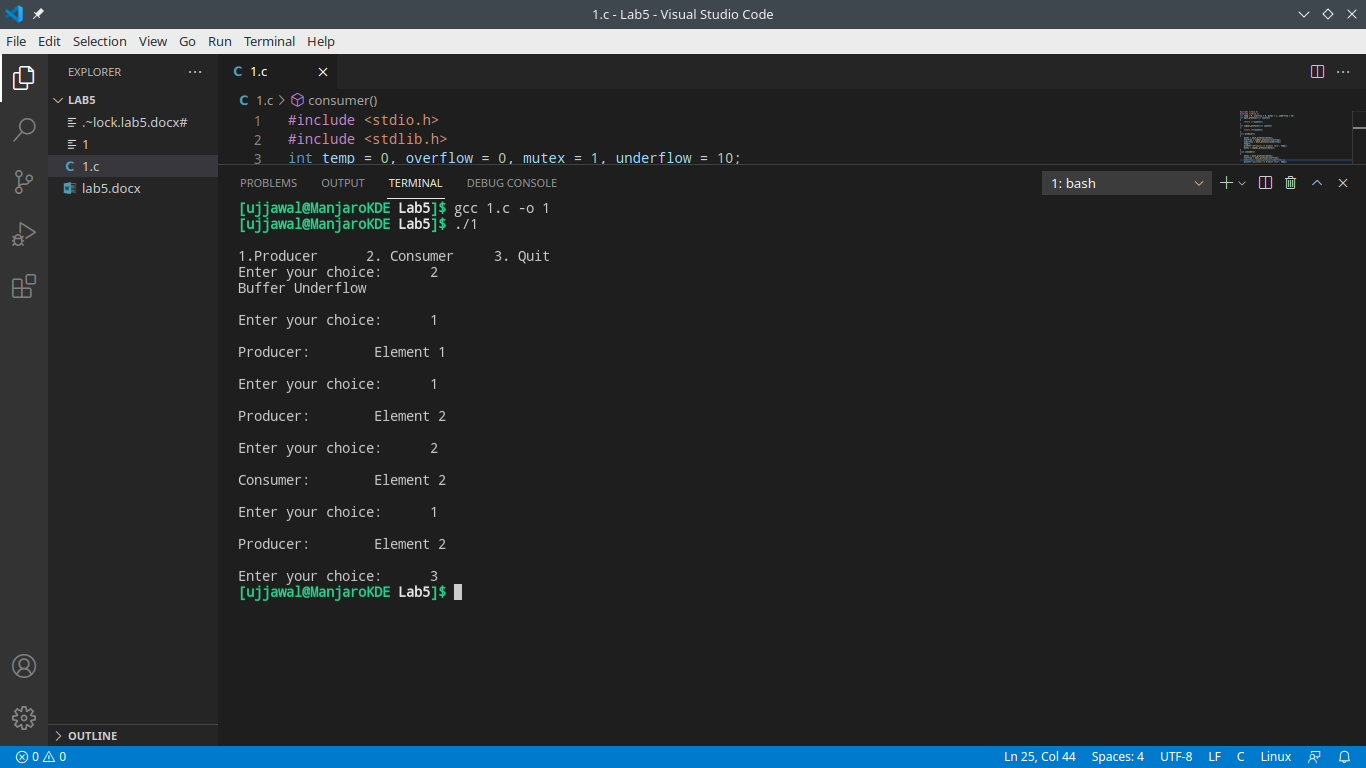
**break;**

**}**

**}**

**return 0;**

**}**



**Question 6**. Demonstrate dead lock in dining philosophers’ problem.

An array of five semaphores, stick[5], for each ofthe five chopsticks.The code for each philosopher looks like:

**while(TRUE)**

**{**

**wait(stick[i]);**

**/\*mod is used because if i=5, nextchopstick is 1 (dining table is circular)\*/**

**wait(stick[(i+1) % 5]);**

**/\* eat \*/**

**signal(stick[i]);**

**signal(stick[(i+1) % 5]);**

**/\* think \*/**

**}**

When a philosopher wants to eat the rice, he willwait for the chopstick at his left and picks up thatchopstick. Then he waits for the right chopstick tobe available, and then picks it too. After eating,heputs both the chopsticks down.But if all five philosophers are hungry simultaneously,and each of them pickup one chopstick, then adeadlock situation occurs because they will be waitingfor another chopstick forever.