**Student Name: Raghav Sharma**

**Student ID: 11813297**

**Email Address: rash.hdr@gmail.com**

**GitHub Link:** **git@github.com:raghavsharma804/OSproject.git**

**Code: Attached in the report.**

**Operating System Simulation Based Assignment**

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**Raghav Sharma**

**Registration number: 11813297**

**Roll No: A16**

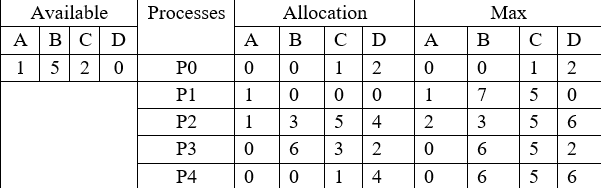
**Section: K18AW**

 **School of Computer Science and Engineering**

Lovely Professional University

Phagwara, Punjab (India)

***Task:***

1. Write a multithreaded program in C that outputs prime numbers. This program should work as follows: The user will run the program and will enter a number on the command line. The program will then create a separate thread that outputs all the prime numbers less than or equal to the number entered by the user.
2. Consider following and Generate a solution in C to find whether the system is in safe state or not?

***Description:***

1. The creation of program for prime number uses two threads in C. First thread will prompt the user to enter the number upto which he wishes to get the prime number for.

The second thread will refer to the input taken by thread 1 and use that to calculate the prime number to that number starting from 1.

It will basically use the concept to multithreading. Advantage of thread here is that system will be able to process two requests from the user at the same time without having individual copy of code in the background.

The code is written in C language and executed in UBUNTU LTS for WINDOWS using -lpthread

*Boundary condition :*

*Number entered by the user must be greater than zero.*

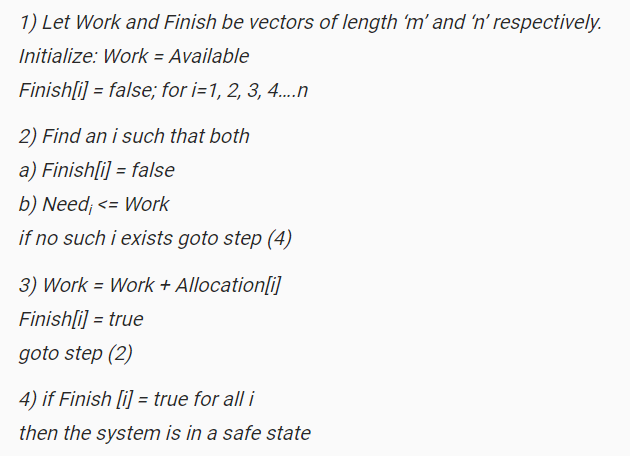
1. Ques 24

To find out whether the system is in safe state or not , we use bankers’ algorithm to calculate the need of each process.

**Banker’s Algorithm**:

**It is used in deadlock avoidance**, such that it tells the system whether a particular system is in safe state or not; whether it will have a deadlock.

1. It is used when there are many instances of resource.
2. If a process wish to request a resource , it must wait for it be vacant.
3. When a process gets all its resource, it will have to terminate and return all the resources back to the system.

Basically Bankers’s algorithm uses Safety algorithm to find out safety of the system

*Overview:*

The allocation table and the maximum table is given. Along with the AVAILABLE TABLE.

Need table is calculated using the formula : Need=Maximum- Allocation.

1. For each process allocation is the resources allocated to it.
2. For each process maximum is the largest number of resources that can be given.
3. For each process, need is the remaining need of resources by the process.

Conditions: Need must be less than the available then only it will be allotted

1. After the need is fulfilled, the allocated resources of that particular process is added to ALLOCATED.

*Boundary conditions:*

*1.User must enter all the details of the allocation and maximum table.*

*2. User must enter the available resources.*

*3. If need is negative then, it will be counted as zero.*

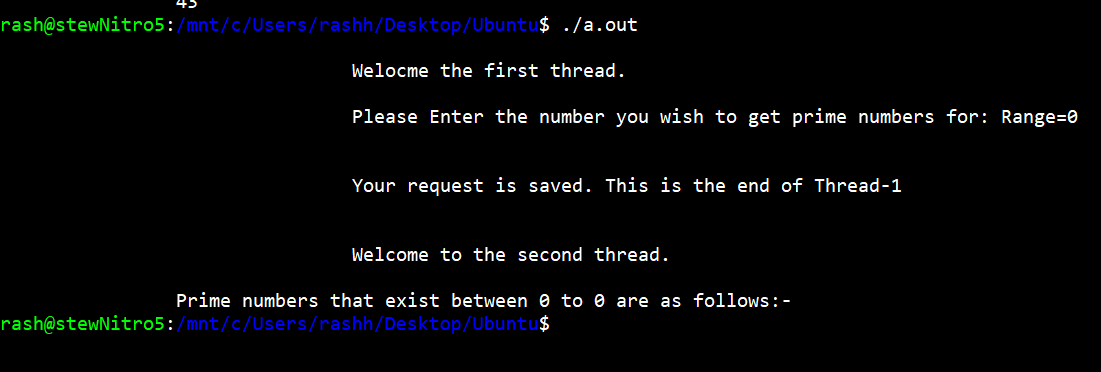
***INPUT/OUTPUT (Multithreading in C; Find out the number of prime numbers within the range of entered digit QUES 3.)***

***Test case 1:***

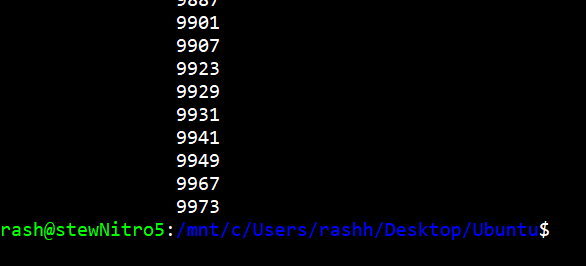
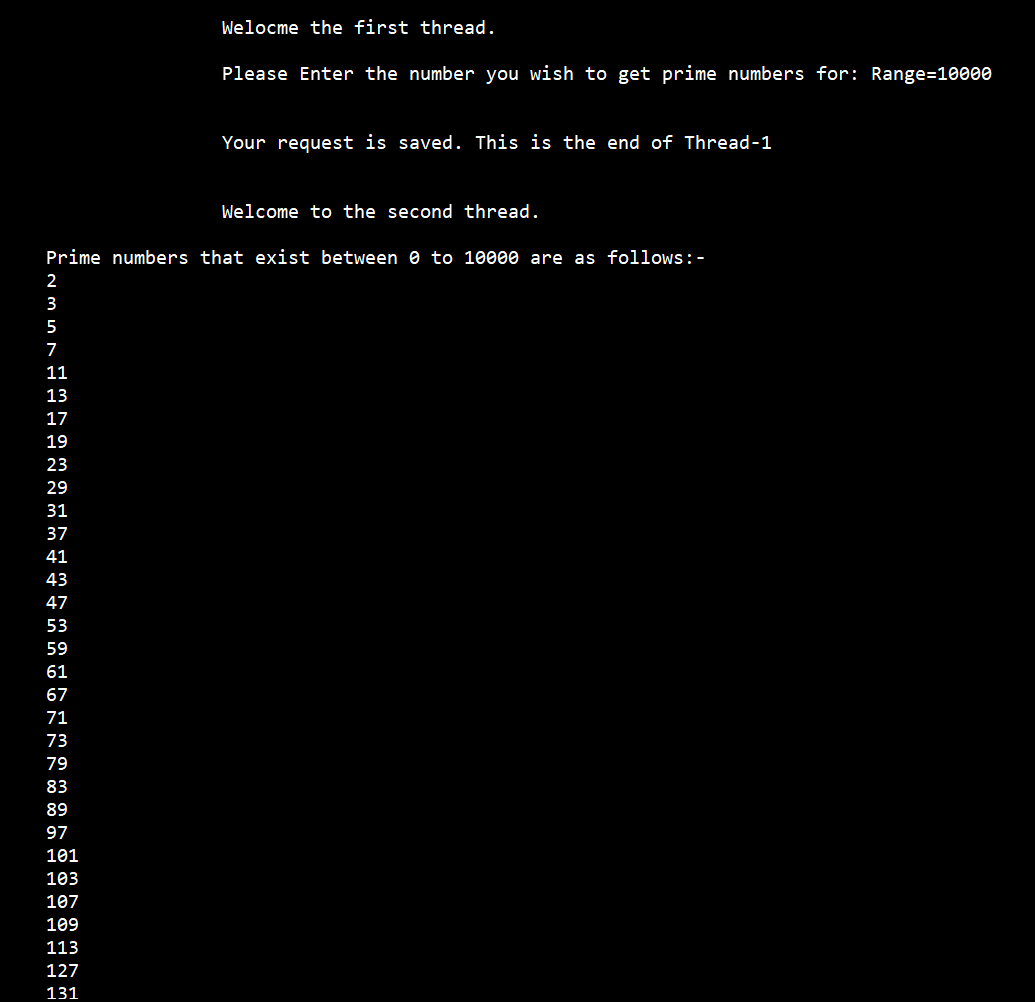
***Entered number is 45***

******

***Test case 2:***

***Entered number is 0*** ******

***Test case 3:***

***Entered number is 10,000.***

***Code for multithreading ques 1:***

*//code written by RAGHAV SHARMA K18AW*

*#include<stdio.h>*

*#include<unistd.h>*

*#include<pthread.h> //contains all the libraries required of threads in C*

*int range; //global variable range declared*

*void \*getrange(void \*value) //start of the first thread*

*{*

*printf("\n\t\t\t\tWelocme the first thread.\n\n\t\t\t\tPlease Enter the number you wish to get prime numbers for: Range="); //asks the user for entering the range*

*scanf("%d",&range); //scans the range integer and saves*

*printf("\n\n\t\t\t\tYour request is saved. This is the end of Thread-1"); //end of thread-1 message*

*return NULL;*

*} //end of thread-1*

*void \*prime(void \*disp) //start of thread 2*

*{*

*int i; //for interating of loop*

*printf("\n\n\n\t\t\t\tWelcome to the second thread.");*

*printf("\n\n\t\tPrime numbers that exist between 0 to %d are as follows:-\n",range);*

*for(i=2;i<=range;i++) //start of for loop for checking prime numbers.*

*{*

*int c=0;*

*for(int j=1;j<=i;j++)*

*{*

*if(i%j==0)*

*{*

*c++;*

*}*

*}*

*if(c==2)*

*{*

*printf("\t\t%d\n",i);*

*}*

*}*

*return NULL;*

*}*

*int main()*

*{*

*pthread\_t threadone; //declaration of Thread-1*

*pthread\_t threadtwo; //declaration of Thread-2*

*pthread\_create(&threadone,NULL,getrange,NULL); //creation of Thread-1:*

*pthread\_join(threadone,NULL);*

*pthread\_create(&threadtwo,NULL,prime,NULL);*

*pthread\_exit(NULL);*

*} //code written by RAGHAV SHARMA K18AW*

***INPUT/OUTPUT (for Banker’s Algorithm ques 24):***

**Test case 1:**

Input used for the test case is:

**PID ALLOCATION MAXIMUM AVAILABLE**

A B C A B C A B C

P1 0 1 1 7 5 3 3 3 2

P2 2 0 0 3 2 2

P3 3 0 2 9 0 2

P4 2 1 1 4 2 2

P5 0 0 2 5 3 3

After calulation, the need table comes out to be

**NEED AVAILABLE**

**7 4 3 3 3 2**

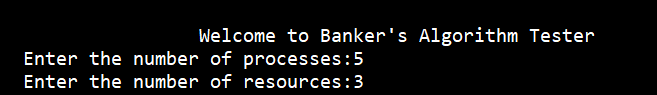
**1 2 2 5 3 2**

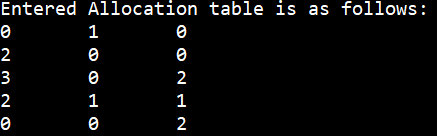
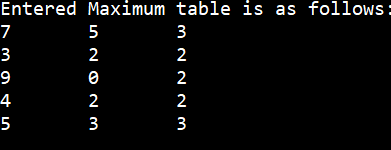
**6 0 0 7 4 3**

**2 1 1 7 4 5**

**5 3 1 7 5 5**

**10 5 7**

**The system is in safe state.**





**Test case 2:**

Input used for the test case is:

**PID ALLOCATION MAXIMUM AVAILABLE**

A B C D A B C D A B C D

P1 0 0 1 2 0 0 1 2 1 5 2 0

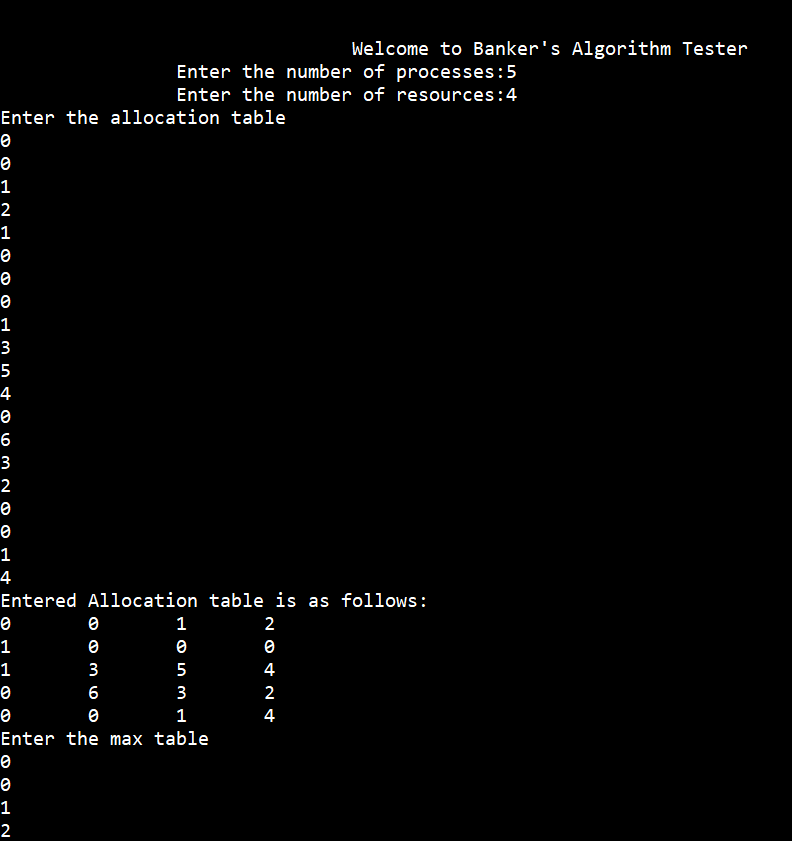
P2 1 0 0 0 1 7 5 0

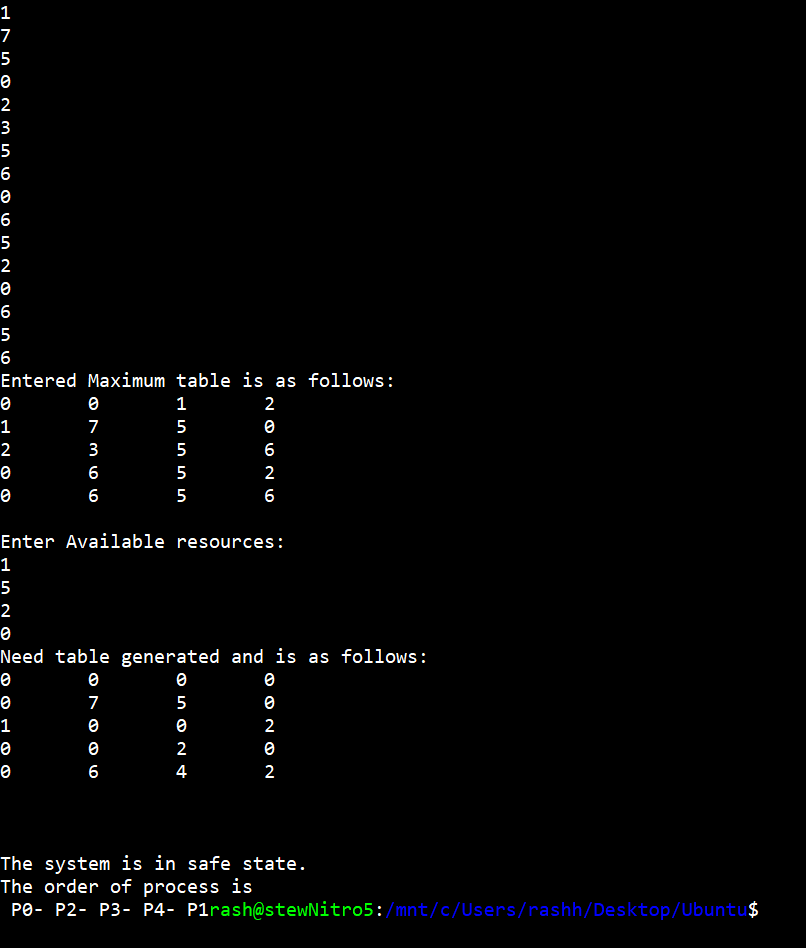
P3 1 3 5 4 2 3 5 6

P4 0 6 3 2 0 6 5 2

P5 0 0 1 4 0 6 5 6

Output Sceen:



******

***Code for Banker’s Algorithm(Ques 24):***

*//code Written by Raghav Sharma*

*#include <stdio.h>*

*#include<unistd.h>*

*int main()*

*{*

*printf("\n\n\n\t\t\t\tWelcome to Banker's Algorithm Tester\n");*

*int i, j, k;*

*int pr,res;*

*int count=0;*

*printf("\t\tEnter the number of processes:");*

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

*printf("\t\tEnter the number of resources:");*

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

*int allocation[pr][res];*

*int maximum[pr][res];*

*int need[pr][res];*

*int avail[4];*

*printf("Enter the allocation table\n");*

*for(int x=0;x<pr;x++)*

*{*

*for(int v=0;v<res;v++)*

*{*

*scanf("%d",&allocation[x][v]);*

*}*

*}*

*printf("Entered Allocation table is as follows:\n");*

*for(int x=0;x<pr;x++)*

*{*

*for(int v=0;v<res;v++)*

*{*

*printf("%d \t",allocation[x][v]);*

*}*

*printf("\n");*

*}*

*printf("Enter the max table\n");*

*for(int x=0;x<pr;x++)*

*{*

*for(int v=0;v<res;v++)*

*{*

*scanf("%d",&maximum[x][v]);*

*}*

*}*

*printf("Entered Maximum table is as follows:\n");*

*for(int x=0;x<pr;x++)*

*{*

*for(int v=0;v<res;v++)*

*{*

*printf("%d \t",maximum[x][v]);*

*}*

*printf("\n");*

*}*

*printf("\nEnter Available resources:\n");*

*for(int z=0;z<res;z++)*

*scanf("%d",&avail[z]); // Available Resources*

*int f[pr], ans[pr], ind = 0;*

*for (k = 0; k < pr; k++) {*

*f[k] = 0;*

*}*

*//NEED TABLE ENTRY*

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

*{*

*for (j = 0; j < res; j++)*

*{*

*need[i][j] = maximum[i][j] - allocation[i][j]; //calculation of need table*

*if(need[i][j]<0) //if need is negetive than it will be counted as zero rather than negetive*

*need[i][j]=0;*

*}*

*}*

*printf("Need table generated and is as follows:\n");*

*for(int x=0;x<pr;x++)*

*{*

*for(int v=0;v<res;v++)*

*{*

*printf("%d \t",need[x][v]);*

*}*

*printf("\n");*

*}*

*int y = 0;*

*for (k = 0; k < 5; k++)*

*{*

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

*{*

*if (f[i] == 0)*

*{*

*int flag = 0; //flag set to 0*

*for (j = 0; j < res; j++) {*

*if (need[i][j] > avail[j])*

*//check if need is greater than available*

*//if yes then skip this process*

*{*

*flag = 1;*

*count=count+1;*

*if(count>pr)*

*//if all the needs are greater than the available resources then the counter will move to label x that will terminate the program.*

*goto x;*

*else*

*break;*

*}*

*}*

*if (flag == 0)*

*{*

*ans[ind++] = i;*

*for (y = 0; y < res; y++)*

*avail[y] += allocation[i][y];*

*f[i] = 1;*

*}*

*}*

*}*

*}*

*printf("\n\n\nThe system is in safe state.\nThe order of process is\n");*

*for (i = 0; i < pr - 1; i++)*

*printf(" P%d-", ans[i]);*

*printf(" P%d", ans[pr - 1]);*

*return (0);*

*x: //label x*

*printf("\n\n\t\t\t\tA deadlock has occured and It is unsafe state!!!\n");*

*return(0);*

*}*

*//code Written by Raghav Sharma*