OS
Programming Assignment 2
Harsh Pandya

Q 64)
The objective of this exercise is to implement a multithreaded solution to find if a given number is a perfect number. N is a perfect number if the sum of all its factors, excluding itself, is N; examples are 6 and 28. The input is an integer, N. The output is true if the number is a perfect number and false otherwise. The main program will read the numbers N and P from the command line. The main process will spawn a set of P threads. The numbers from 1 to N will be partitioned among these threads so that two threads do not work on the name number. For each number in this set, the thread will determine if the number is a factor of N. If it is, it adds the number to a shared buffer that stores factors of N. The parent process waits till all the threads complete. Use the appropriate synchronization primitive here. The parent will then determine if the input number is perfect, that is, if N is a sum of all its factors and then report accordingly.

Soln.
Here, in the below program the number to be tested ‘A’ &
The number of threads to be used to test the number ‘P’ are both passed as Command Line Arguments.
Using the variable ‘x’ we equally divide the work among all the threads.
If the work cannot be equally divided the extra work is done by the last thread that is created. Also the computation is made faster by using variable ‘b’ to calculate values up to Square root of N.
And thread_join is used for synchronization and waits for a particular thread to finish its execution.
Each thread checks whether the numbers assigned to the are factors of A. If yes, they are stored in a Buffer – ‘Factors’ and the pointer ‘space’ is updated.
Also Mutex are used to avoid race conditions when the threads try to access the shared memory.
When the thread execution is finished the main calculates the Sum of all factors Summation to verify if the given number is a perfect number.

EXECUTION STEPS:

1. Compile the program in the Terminal:
   gcc –pthread filename.c –lm –o filename

2. Run the program in the Terminal with command line arguments:
   ./filename N P

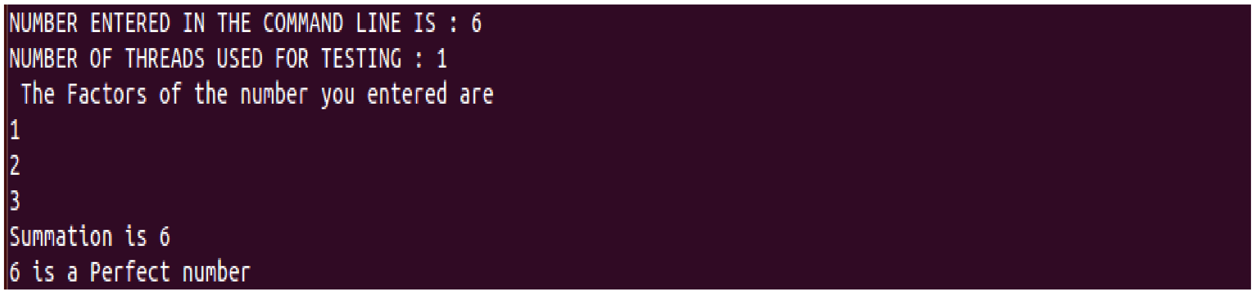
Program:

```
#include <stdio.h>
#include <pthread.h>
#include <math.h>
#include <stdlib.h>

int x,b;                                          //Variable declaration
int space=0;
int Factors[100];                                 // Buffer for storing factors
int A,P;
```

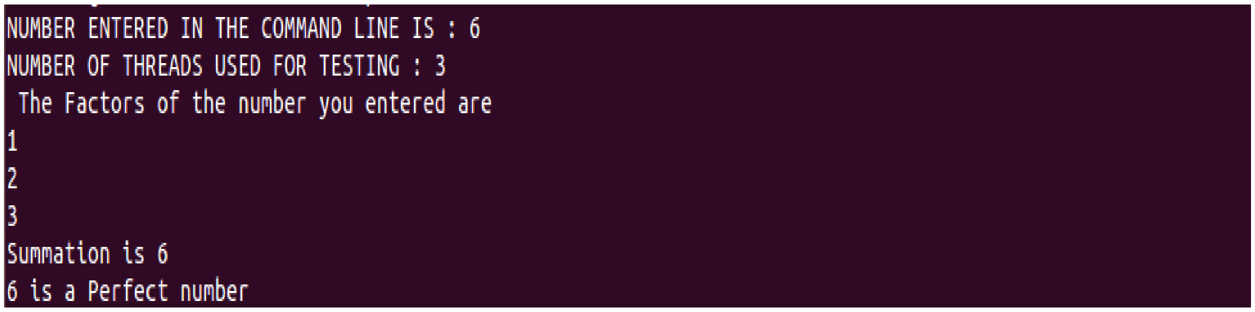
```
pthread_mutex_t region_mutex = PTHREAD_MUTEX_INITIALIZER;

//Initialize a Mutex

void* P_NO(void* no_thread);
int main(int argc, const char * argv[])
{
	A = atoi(argv[1]);				//Takes the first argument and converts it to integer
	P = atoi(argv[2]);				//Takes the second thread argument and converts it to int
	printf("NUMBER ENTERED IN THE COMMAND LINE IS : %d \n",A);
	b=sqrt(A);					//Quicker calculation we use SquareRoot of N
	printf("NUMBER OF THREADS USED FOR TESTING : %d \n",P);
	b=A;
	pthread_t T[P];				//Stores the ThreadID
	int i;
	int Summation=0;				//To calculate the sum of Factors
	for(i=0;i<P;i++)
	{
		pthread_create(&T[i],NULL,P_NO,(void *)i);		// 'P' Threads are created
	}
	for(i=0;i<P;i++)
	{
		pthread_join(T[i],NULL);
		//Using Sychronization and merging it with the parent thread
	}
	int c,j,k;
	printf(" The Factors of the number you entered are \n" );
	for (i = 0; i < space; i++)
	{
		for (j = i + 1; j < space;)
		{
			if (Factors[j] == Factors[i]) //removes double repeated factors
			{
				for (k = j; k < space; k++)
				{
					Factors[k] = Factors[k + 1];
				}
				space--;
			}
			else
			j++;
		}
	}
	for(c=0;c<space;c++)				//To evaluate Summation of all Factors
	{
		Summation+=Factors[c];
		printf("%d \n",Factors[c]);
	}
	printf("Summation is %d \n",Summation);	//To check if Sum Of Factors = Number
	if(Summation==A)
	{
		printf("%d is a Perfect number \n", A);
	}
	else
	{
		printf("%d is NOT a Perfect Number \n", A);
	}
	return 0;
}
```

```
void* P_NO(void* no_thread)
{
        x=b/P;                                  // work to be done by each thread
        int i;
        int Begin,Last;                         //Declaring the variables to save the boundaries
        int id_thread= (int) no_thread;         //Typecasting
        if(id_thread==(P-1))                    //Calculating the Begin and Last boundaries for thread
        {
                Begin=id_thread*x+1;
                Last=b;
        }
        else
        {
                Begin=id_thread*x+1;
                Last=Begin+x;
        }
        //Each thread calculates factors of A
        for(i=Begin;i<Last;i++)
        {
                if(A%i==0)
                {
                        pthread_mutex_lock(&region_mutex);
                        {
                                if(i==1)
                                {
                                        Factors[space]=1;       //saves result in Buffer
                                        space++;                //pointer gets incremented.
                                }
                                else
                                {
                                        Factors[space]=i;       //for quicker results
                                        space++;
                                        Factors[space]=A/i;
                                        space++;
                                }
                        }

                        pthread_mutex_unlock(&region_mutex);
                }
        }

        return 0;
}
```

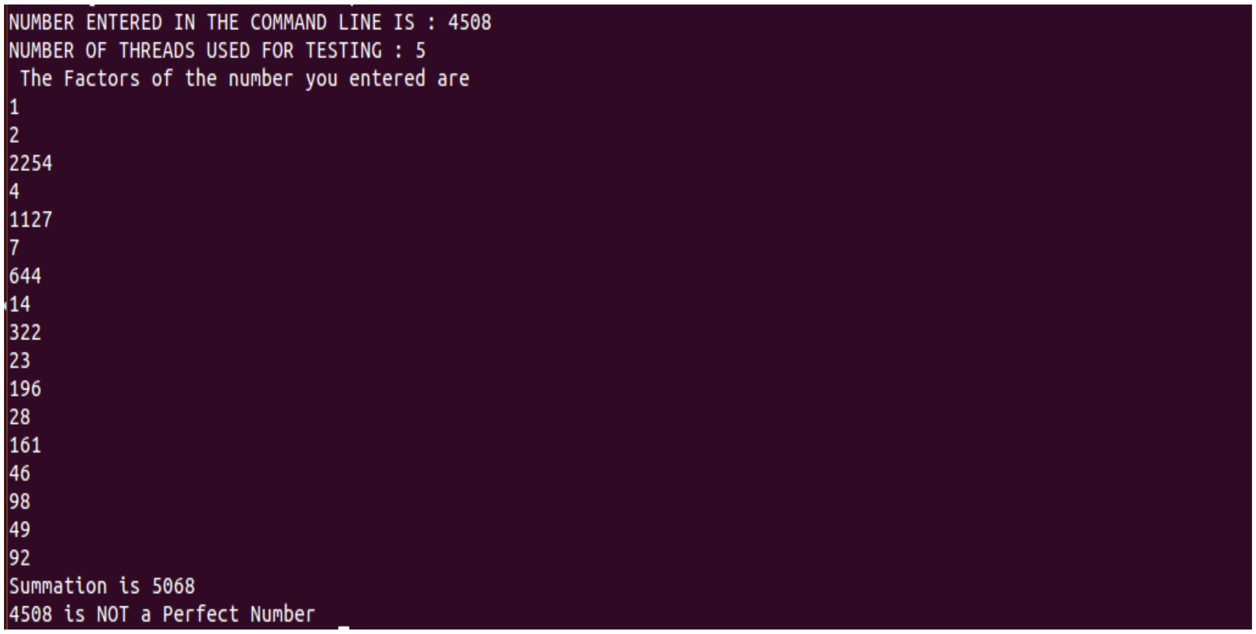
OUTPUT:



```
NUMBER ENTERED IN THE COMMAND LINE IS : 6
NUMBER OF THREADS USED FOR TESTING : 1
 The Factors of the number you entered are
1
2
3
Summation is 6
6 is a Perfect number
```



```
NUMBER ENTERED IN THE COMMAND LINE IS : 6
NUMBER OF THREADS USED FOR TESTING : 3
 The Factors of the number you entered are
1
2
3
Summation is 6
6 is a Perfect number
```



```
NUMBER ENTERED IN THE COMMAND LINE IS : 4508
NUMBER OF THREADS USED FOR TESTING : 5
 The Factors of the number you entered are
1
2
2254
4
1127
7
644
14
322
23
196
28
161
46
98
49
92
Summation is 5068
4508 is NOT a Perfect Number
```