**National University of Computer & Emerging Sciences, Karachi Department of Computer Sciences**



**Quiz-2(Spring-2024)**

|  |  |
| --- | --- |
| **Course No:** | **Course Name: Operating Systems** |
| **Instructor Name :M.Minhal Raza** | |
| **Student Roll No:** | **Section: BSE-4A** |

**"If there is something, you don’t know today. You will surely learn afterwards. Life is not an exam hall."**

**BEST OF LUCK!**

**Time Allowed**: 25 minutes. **Total Marks**: 100

Instructions:

* Your Student ID must be written on the paper.

**Question # 1: NADRA Pakistan DHA Office** who schedules all her citizens for 30 minutes services each counter. Some of the counters take more or less than 30 minutes depending on the type of service work to be done. The following summary shows the various categories of work, and the time actually needed to complete the work

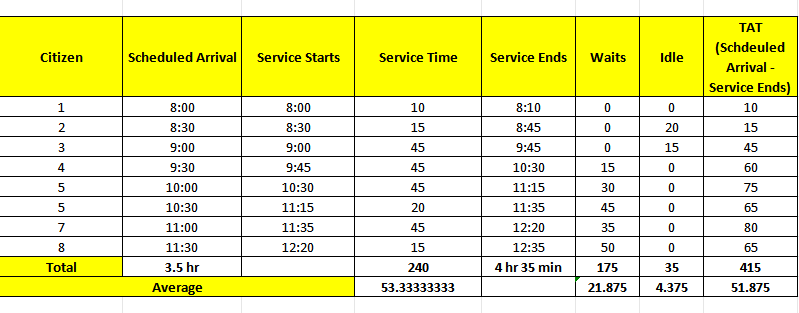
|  |  |  |  |
| --- | --- | --- | --- |
| Category | Service Time | Random Customer CNIC Numbers Ending With | Per Day Customer Service |
| Waiting in Line | 45 mints | 00-39 | 40 |
| Token Acquires | 10 mints | 40-54 | 15 |
| Service on Counter, (Snap,Application Form Filing,Data Entering,Biometric,Signing,Payment,Investigation,Application Form Printout) | 20 mints | 55-69 | 15 |
| Application Form Attestation | 15 mints | 70-79 | 10 |
| Application Form Submission and Exit | 15 mints | 80-99 | 20 |
| Total Customer Per Day | | | 100 |

Now, Simulate the scheduling using **FCFS** for four hours and find out the average waiting time for the citizens and the idleness of the counter. Assume that all the citizens show up at the franchise at exactly their scheduled time; let us consider the arrival time for the first citizens as 8.00 AM. You may use the following random numbers of citizens having ending CNIC Number for handling the above problem during the simulation [40, 82, 11, 34, 25, 66, 17, 79].

Now, the first citizens is arriving at exactly 8.00 AM. Each citizen's average serving time is 30 minutes. According to the problem, we have to simulate the situation for the next four hours, equal to 240 minutes. So, the counter will serve the citizens till 12.00 PM. The maximum number of

citizens can be allotted for the service is 8. Kindly see the below table carefully





**Question-2**

Write a multi-threaded program to find if a given number is a perfect number. A number N is a perfect number if the sum of all its factors, excluding itself, is N; examples are 6(1+2+3) and 28 (1+2+4+7+14). 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. 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.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define MAX\_THREADS 100

#define MAX\_FACTORS 1000

// Global variables

int factors[MAX\_FACTORS];

int num\_factors = 0;

int n;

int p;

void \*find\_factors(void \*arg) {

int thread\_id = \*((int \*)arg);

for (int i = thread\_id + 1; i <= n; i += p) {

if (n % i == 0) {

factors[num\_factors++] = i;

}

}

return NULL;

}

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

if (argc != 3) {

printf("Usage: %s <N> <P>\n", argv[0]);

return 1;

}



n = atoi(argv[1]);

p = atoi(argv[2]);

if (n <= 0 || p <= 0) {

printf("N and P must be positive integers.\n");

return 1;

}

if (num\_factors >= MAX\_FACTORS) {

printf("Number of factors exceeds maximum allowed.\n");

return 1;

}

pthread\_t threads[MAX\_THREADS];

int thread\_ids[MAX\_THREADS];

// Create threads

for (int i = 0; i < p; i++) {

thread\_ids[i] = i;

pthread\_create(&threads[i], NULL, find\_factors, &thread\_ids[i]);

}

// Join threads

for (int i = 0; i < p; i++) {

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

}

// Check if the number is perfect

int sum = 0;

for (int i = 0; i < num\_factors; i++) {

sum += factors[i];

}

if (sum == n) {

printf("%d is a perfect number.\n", n);

} else {

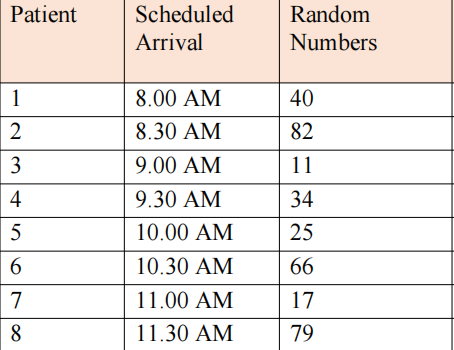
printf("%d is not a perfect number.\n", n);

}

return 0;

}

**Appendix for Question-1:**





1 OF 2