Hau Tao
005393943
Homework 3

1. The previous four runs, from oldest to most recent are 40, 20, 40, and 15 msec

Suppose the initial predicted time $T_0 = t_0$=40 ms.

$t_0 = 40$, $t_1 = 20$, $t_2 = 40$, $t_3 = 15$

We have : $T_4 = t_3/2 + T_3/2 = t_3/2 + t_2/4 + T_2/4 = t_3/2 + t_2/4 + t_1/8 + T_1/8$

$= t_3/2 + t_2/4 + t_1/8 + t_0/16 + T_0/16 = 15/2 + 40/4 + 20/8 + 40/16 + 40/16$

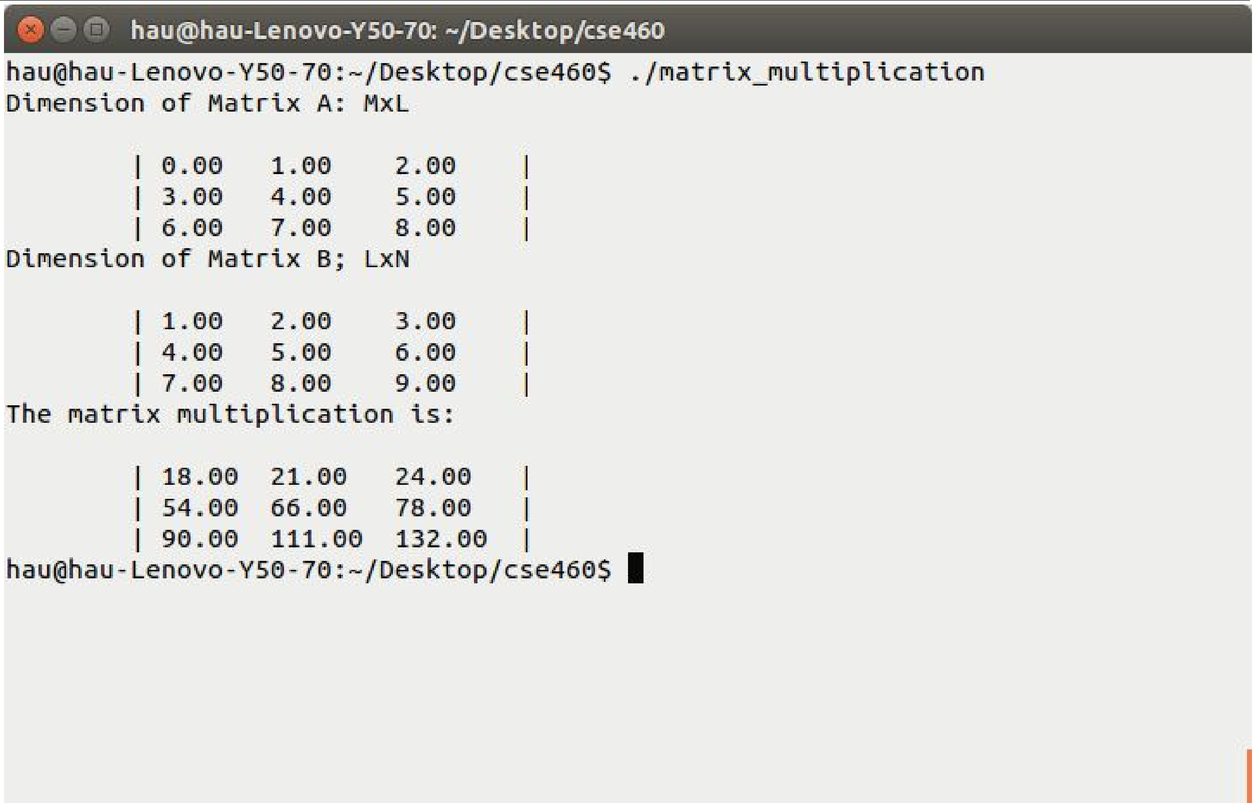
= 7.5 + 10 + 2.5+2.5+2.5= 25 ms

2. % of CPU time 'wasted' = T/(T+S):
   a. Q = infinity -> CPU efficiency = T / (T+S)
   b. Q > T -> no effect on CPU efficiency = T / (T+S)
   c. S < Q <T -> The time each process takes should be Q-> CPU efficiency = Q /(Q+S)
   d. Q nearly 0 -> the time each process takes is nearly 0-> CPU efficiency = 0/(0+S) = 0

3.

```
// matrix_multiplication.cpp
// Use thread to do matrix multiplication 3x3, could enhance more dimension
// @Author: Hau Tao

#include <stdio.h>
#include <stdlib.h>
#include <SDL/SDL.h>
#include <iostream>

/* We must include SDL_thread.h separately. */
#include <SDL/SDL_thread.h>
#include <vector>

using namespace std;

const int M = 3;
const int L = 3;
const int N = 3;

//shared variables
double A[M][L];
double B[L][N];
double C[M][N];
double sum;

/* This function is a thread entry point. */
typedef struct {
  int row;
  int column;
} ThreadData;
int dotProduct ( void *data )
{

       ThreadData  *tdata= (ThreadData*) data;
       sum = 0.0;
       for ( int i = 0; i < L; i++ )
       sum += A[tdata->row][i] * B[i][tdata->column];
       C[tdata->row][tdata->column] = sum;


       return 0;
}
void initMatrixA(double a[][L], int M)
{
  //some arbitrary data
  double value = 0.0;
  for ( int i = 0; i < M; i++ ) {
       for(int j=0; j< L; j++)
       a[i][j] = value++;

  }
}
void initMatrixB(double b[][N], int L)
{
  //some arbitrary data
  double value = 1.0;
  for ( int i = 0; i < L; i++ ) {
       for(int j=0; j< N; j++)
       b[i][j] = value++;

  }
}



void print( const double a[][L], int M)
{
  int i, j;
  for (i = 0; i < M; i++) {
       printf("\n\t| ");
       for (j = 0; j < L; j++){
       printf("%.2f", a[i][j]);
       printf("\t ");
       }
       printf("|");
  }
}

int main()
{
       cout << "Dimension of Matrix A: MxL\n ";
       initMatrixA(A, M);
       print(A,M);
```

```
        cout <<"\nDimension of Matrix B; LxN\n";
        initMatrixB(B,L);
        print(B,L);
        cout << endl;
        ThreadData *data = (ThreadData*) malloc(sizeof(ThreadData));
        SDL_Thread *sumThread;
        int i, j;
        for (i =0; i < M; i++){
        for(j=0;j < N; j++){
        data->row = i;
        data->column =j;
        sumThread = SDL_CreateThread( dotProduct, (void*) data);
        if ( sumThread == NULL )
                cout << "\nSDL_CreateThread failed: " <<  SDL_GetError() << endl;
        else{
                int returnValue;
                SDL_WaitThread( sumThread, &returnValue);

        }

        }
        }
        cout <<"The matrix multiplication is:\n";
        print(C,M);
        cout << endl;
        return 0;
}
```



```
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ./matrix_multiplication
Dimension of Matrix A: MxL

        | 0.00   1.00    2.00    |
        | 3.00   4.00    5.00    |
        | 6.00   7.00    8.00    |
Dimension of Matrix B; LxN

        | 1.00   2.00    3.00    |
        | 4.00   5.00    6.00    |
        | 7.00   8.00    9.00    |
The matrix multiplication is:

        | 18.00  21.00   24.00   |
        | 54.00  66.00   78.00   |
        | 90.00  111.00  132.00  |
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$
```

4.
sample output

```
hau@hau-Lenovo-Y50-70:~/Desktop/cse460$ ./readers_writers

This is reader 1 thread
Counter: 0 was read.

This is reader 2 thread
Counter: 0 was read.

This is reader 3 thread
Counter: 0 was read.

This is reader 4 thread
Counter: 0 was read.

This is reader 5 thread
Counter: 0 was read.

This is reader 6 thread
Counter: 0 was read.

This is reader 7 thread
Counter: 0 was read.

This is reader 8 thread
Counter: 0 was read.

This is reader 9 thread
Counter: 0 was read.

This is reader 10 thread
Counter: 0 was read.

This is reader 11 thread
Counter: 0 was read.

This is reader 12 thread
Counter: 0 was read.

This is reader 13 thread
Counter: 0 was read.

This is reader 14 thread
Counter: 0 was read.
```

```
This is reader 15 thread
Counter: 0 was read.

This is reader 16 thread
Counter: 0 was read.

This is reader 17 thread
Counter: 0 was read.

This is reader 18 thread
Counter: 0 was read.

This is reader 19 thread
Counter: 0 was read.

This is reader 20 thread
Counter: 0 was read.

This is writer 1 thread
Writing: 1 to the file


This is writer 2 thread
Writing: 2 to the file


This is writer 3 thread
Writing: 3 to the file


This is reader 1 thread
Counter: 3 was read.

This is reader 2 thread
Counter: 3 was read.

This is reader 3 thread
Counter: 3 was read.

This is reader 4 thread
Counter: 3 was read.

This is reader 5 thread
Counter: 3 was read.

This is reader 6 thread
Counter: 3 was read.
```

```
This is reader 7 thread
Counter: 3 was read.

This is reader 8 thread
Counter: 3 was read.

This is reader 9 thread
Counter: 3 was read.

This is reader 10 thread
Counter: 3 was read.

This is reader 11 thread
Counter: 3 was read.

This is reader 12 thread
Counter: 3 was read.

This is reader 13 thread
Counter: 3 was read.

This is reader 14 thread
Counter: 3 was read.

This is reader 15 thread
Counter: 3 was read.

This is reader 16 thread
Counter: 3 was read.

This is reader 17 thread
Counter: 3 was read.

This is reader 18 thread
Counter: 3 was read.

This is reader 19 thread
Counter: 3 was read.

This is reader 20 thread
Counter: 3 was read.

This is writer 1 thread
Writing: 4 to the file
```

```
This is writer 2 thread
Writing: 5 to the file


This is writer 3 thread
Writing: 6 to the file


This is reader 1 thread
Counter: 6 was read.

This is reader 2 thread
Counter: 6 was read.

This is reader 3 thread
Counter: 6 was read.

This is reader 4 thread
Counter: 6 was read.

This is reader 5 thread
Counter: 6 was read.

This is reader 6 thread
Counter: 6 was read.

This is reader 7 thread
Counter: 6 was read.

This is reader 8 thread
Counter: 6 was read.

This is reader 9 thread
Counter: 6 was read.

This is reader 10 thread
Counter: 6 was read.

This is reader 11 thread
Counter: 6 was read.

This is reader 12 thread
Counter: 6 was read.

This is reader 13 thread
Counter: 6 was read.
```

```
This is reader 14 thread
Counter: 6 was read.

This is reader 15 thread
Counter: 6 was read.

This is reader 16 thread
Counter: 6 was read.

This is reader 17 thread
Counter: 6 was read.

This is reader 18 thread
Counter: 6 was read.

This is reader 19 thread
Counter: 6 was read.

This is reader 20 thread
Counter: 6 was read.

This is writer 1 thread
Writing: 7 to the file


This is writer 2 thread
Writing: 8 to the file


This is writer 3 thread
Writing: 9 to the file


This is reader 1 thread
Counter: 9 was read.

This is reader 2 thread
Counter: 9 was read.

This is reader 3 thread
Counter: 9 was read.

This is reader 4 thread
Counter: 9 was read.

This is reader 5 thread
Counter: 9 was read.
```

```
This is reader 6 thread
Counter: 9 was read.

This is reader 7 thread
Counter: 9 was read.

This is reader 8 thread
Counter: 9 was read.

This is reader 9 thread
Counter: 9 was read.

This is reader 10 thread
Counter: 9 was read.

This is reader 11 thread
Counter: 9 was read.

This is reader 12 thread
Counter: 9 was read.

This is reader 13 thread
Counter: 9 was read.

This is reader 14 thread
Counter: 9 was read.

This is reader 15 thread
Counter: 9 was read.

This is reader 16 thread
Counter: 9 was read.

This is reader 17 thread
Counter: 9 was read.

This is reader 18 thread
Counter: 9 was read.

This is reader 19 thread
Counter: 9 was read.

This is reader 20 thread
Counter: 9 was read.

This is writer 1 thread
```

```
Writing: 10 to the file


This is writer 2 thread
Writing: 11 to the file


This is writer 3 thread
Writing: 12 to the file
```

```
/*
  readers_writers.cpp
  Compile:  g++ -o  readers_writers readers_writers.cpp -lSDL -lpthread
  Execute:  ./readers_writers
*/

#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <string>
#include <exception>
using namespace std;

SDL_bool condition = SDL_FALSE;
SDL_mutex *mutex;
SDL_cond *readerQueue;   //condition variable
SDL_cond *writerQueue;   //condition variable

int readers = 0;
int writers = 0;
int num = 1;
void read(){
       try
       {
        string line,new1;
       ifstream fin("counter.txt");
       string data;


       while (getline (fin,line))
        {
                       new1=line;
        }
        cout  <<"Counter: " << new1<< " was read.\n";
       fin.close(); //close file


       }
       catch(exception &e)
       {
       cerr<<e.what()<<endl; //display any error that may occur

       }

}

void write(){
  try
       {
       ofstream fout("counter.txt",ios::app); //open file for writing and append to it
       cout <<"Writing: " << num;
       fout << num << endl;
       num++;
       cout <<" to the file \n"<< endl;
       fout.close(); //close the file

       }
       catch(exception &e)
       {
       cerr<<e.what(); //write any exceptions that may occur when trying to write to file

       }

}

int reader ( void *data )
{
  SDL_LockMutex ( mutex );
  while ( !(writers == 0) )
       SDL_CondWait ( readerQueue, mutex );

  readers++;
  printf("\nThis is %s thread\n", (char *) data );
  SDL_UnlockMutex ( mutex );
  read();
  SDL_Delay ( rand() % 3000);
  SDL_LockMutex ( mutex );
  //printf("\nThis is %s thread\n", (char *) data );
  if ( --readers == 0 )
       SDL_CondSignal ( writerQueue );
  SDL_UnlockMutex ( mutex );
}

int writer ( void *data )
{
  SDL_LockMutex(mutex);
  while ( !( (readers == 0) && (writers == 0) ) )
       SDL_CondWait ( writerQueue, mutex );

  writers++;
  printf("\nThis is %s thread\n", (char *) data );
  SDL_UnlockMutex ( mutex );
  write();
  SDL_Delay ( rand() % 3000);
  SDL_LockMutex ( mutex );
  writers--;          //only one writer at one time
  SDL_CondSignal ( writerQueue );
  SDL_CondBroadcast ( readerQueue );
  SDL_UnlockMutex ( mutex );
```

```
}

int main ()
{
  SDL_Thread *idr[20], *idw[3];                    //thread identifiers
  char *rnames[] = { "reader 1", "reader 2", "reader 3",
  "reader 4", "reader 5", "reader 6", "reader 7", "reader 8", "reader 9", "reader 10",
  "reader 11", "reader 12", "reader 13",
  "reader 14", "reader 15", "reader 16", "reader 17", "reader 18", "reader 19", "reader
20"}; //names of threads
  char *wnames[] = { "writer 1", "writer 2", "writer 3" }; //names of threads

  mutex = SDL_CreateMutex();
  readerQueue = SDL_CreateCond();
  writerQueue = SDL_CreateCond();
  int i,j;
  while(1){
       for (  i = 0; i < 20; i++ ){
       idr[i] = SDL_CreateThread ( reader, rnames[i] );
       SDL_Delay ( rand() % 3000);
       }
       for (  j = 0; j < 3; j++ ){
       idw[j] = SDL_CreateThread ( writer, wnames[j] );
       SDL_Delay ( rand() % 3000);
       }
  }

  i=j=0;

  for ( i = 0, j=0; i < 20 or j<3; i++, j++ ){
       SDL_WaitThread ( idr[i], NULL );
       SDL_WaitThread ( idw[j], NULL );
  }

  SDL_DestroyCond ( readerQueue );
  SDL_DestroyCond ( writerQueue );
  SDL_DestroyMutex ( mutex );
  return 0;
}
```