UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 8 Examination in Engineering: November 2017

Module Number: EE8203 **Module Name: High Performance Computing**

[Three Hours]

[Answer **all questions**, each question carries 10 marks]

Q1 a) The following program shows the Peterson's algorithm for two processes. Explain the Peterson's algorithm using following source code.

```
bool flag[2] = {false, false}; // shared variables
int turn;                      // shared variables
```

Process One	Process Two
`// Process one`	`//Process Two`
`flag[0] = true;`	`flag[1] = true;`
`turn = 1;`	`turn = 0;`
`while (flag[1] && turn == 1)`	`while (flag[0] && turn == 0)`
`{`	`{`
`// Section 1`	`// Section 3`
`}`	`}`
`// Section 2`	`//Section 4`
`flag[0] = false;`	`flag[1] = false;`

[2 Marks]

b) Dining philosopher problem is classical problem used in concurrent algorithm design. According to this classical problem five silent philosophers sit around a table. There is a fork placed between each pair of adjacent philosophers. Each philosopher is having a plate with food. A philosopher needs two forks to eat. These philosophers think for a while and eat for a while.

i) How can the philosophers end up with livelock state?
ii) Briefly explain what is resource starvation related to the above classical problem.

[4 Marks]

c) With the MapReduce implementation of Apache Software Foundation, what is the default input key-value pairs for the map function when processing text files.

[2 Marks]

d) What are the two default operations which are applied on the input key-value pairs of the reduce function in MapReduce?

[2 Marks]

Q2 a) What is the meaning of shared memory model which is used with OpenMP?

[1 Mark]

b) Explain the fork-join model of OpenMP using a diagram.
Note: Mark the master thread

[1 Mark]

c) Use the following program to answer the questions below

```
#include <omp.h>
#define N 1000
#define CHUNKSIZE 100

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

int i, chunk;
float a[N], b[N], c[N];

for (i=0; i < N; i++)
   a[i] = b[i] = i * 1.0;
chunk = CHUNKSIZE;

#pragma omp parallel shared(a,b,c,chunk) private(i)
{
   #pragma omp for schedule(dynamic,chunk) nowait
   for (i=0; i < N; i++)
      c[i] = a[i] + b[i];
}
}
```

i) What is the purpose of using **nowait** key word in the above OpenMP program?

ii) What is the difference between **shared(a,b,c,chunk)** vs **private(i)**?

[2 Marks]

d) Write a program using OpenMP to threshold the following grayscale image of size 100 x 100. Intensity values are varying from 0 to 99.
Note: You can represent the image by a two dimensional array of fixed size.
Function to threshold the image is given below.

0	0	0	0	0	0	..	..	0	0
1	1	1	1	1	1	..	..	1	1
2	2	2	2	2	2	..	..	2	2
..	..								..
..	..								..
..	..								..
..	..								..
99	99	99	99	99	99	..	..	99	99

```
int threshold(int pixelValue){
   if(pixelValue>80){
            return 1;
   }else{
            return 0;
   }
}
```

[2 Marks]

e) Mutex variable in POSIX threads is declared as
`pthread_mutex_t mymutex;`
What is the purpose of using such mutex variables in POSIX threads?

[2 Marks]

f) What is the main difference between OpenMP and POSIX threads when it comes to the thread creation and termination?

[2 Marks]

Q3 a) Write the output of the following program.

```
using System;
using System.Threading;
namespace HPC
{
    public static class Program
    {
        public static void ThreadMethod()
        {
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine("ThreadProc: {0}", i);
                Thread.Sleep(1000);
            }
        }
        public static void Main()
        {
            Console.WriteLine("I am from Main Thread");
            Thread t = new Thread(new ThreadStart(ThreadMethod));
            t.IsBackground = true;
            t.Start();
        }
    }
}
```

[2 Marks]

b) What is the purpose of `join()` method of the C# thread class?

[2 Marks]

c) What is the main advantage of using a thread pool in C# programming language?

[2 Marks]

d) Fill the missing parts of the following C# program to do the following.

i) No of iterations in the for-loop must be given as a parameter to the thread method.

ii) Set the parameter of the thread method as 6, where the for-loop will execute for six iterations.

iii) Join the created thread with the main thread in the main function.

```
public static void ThreadMethod(.....)
{
    for (int i = 0; i < .....  ; i++)
    {
        Console.WriteLine("ThreadProc: {0}", i);
        Thread.Sleep(0);
    }
}
```

```
public static void Main()
{
    Thread t = new Thread(new ParameterizedThreadStart(ThreadMethod));
    ……
    ……..
}
```

[4 Marks]

Q4 a) Complete the following CUDA C program to add 2 matrixes of size NxN. Note: In the main function assume the matrixes are already declared and memory is transferred to the GPU card, write only the two lines related to the grid dimension and the kernel launch.

```
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
……..
……..
……..
……..
}

int main()
{

   // matrixes A, B and C already declared
    dim3 threadsPerBlock(16, 16);
    ……..
    ……..
}
```

[3 Marks]

b) What is the purpose of `cudaMemcpy()` function in CUDA C programming model?

[1 Mark]

c) Draw the memory hierarchy of CUDA. Include local memory, shared memory and global memory.

[2 Marks]

d) List two limitations when programming with Graphics Processing Units.

[2 Marks]

e) Is it possible to execute CPU code and GPU code concurrently? Give reasons to your answer.

[2 Marks]

Q5 a) Draw the message-passing parallel programming model of MPI.

[2 Marks]

b) What is the functionality of `MPI_Ssend()` method in MPI?

[2 Marks]

c) Describe the output of the following MPI program using a diagram when world_size=2.

```
#include "mpi.h"
#include <stdio.h>
void main(int argc, char *argv[])
{
  int numTask, rank, dest, source;
  int no1=-5, no2=2, tag=1;
  MPI_Status Stat;

  MPI_Init(&argc,&argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  if(rank==0){
   dest=1;
   source=1;
   MPI_Send(&no2,1,MPI_INT,dest,tag, MPI_COMM_WORLD);
   MPI_Recv(&no2,1,MPI_INT,source,tag,MPI_COMM_WORLD,
&Stat);
  }else if(rank==1){
   dest=0;
   source=0;
   MPI_Recv(&no1,1,MPI_INT,source,tag,MPI_COMM_WORLD,
&Stat);
   no1 = no1*(rank+1);
   MPI_Send(&no1,1,MPI_INT,dest,tag, MPI_COMM_WORLD);
  }
  printf("The rank is:%d\n",rank);
  printf("The value of no1 is:%d\n",no1);
  printf("The value of no2 is:%d\n",no2);
}
```

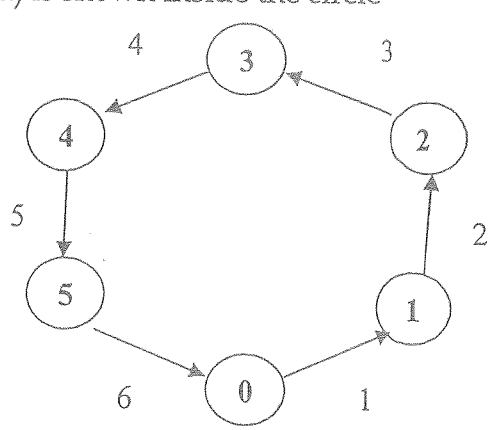
[3 Marks]

d) Write a MPI program using the C programming language which does the following.

i) Message passing starts at task 0 by sending the number one to task 2 and ends when task 0 receive 6 from task 5.

ii) Arrows shows the messages passed and each task increment the received by 1 and send it to the next task.

Note: Task ID (rank) is shown inside the circle



[3 Marks]