UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 8 Examination in Engineering: November / December 2016

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

[Three Hours]

[Answer all questions, each question carries 10 marks]

Q1 a) What is the main purpose of Peterson's algorithm?

[2 Marks]

b) The following program shows the Peterson's algorithm for two processes.

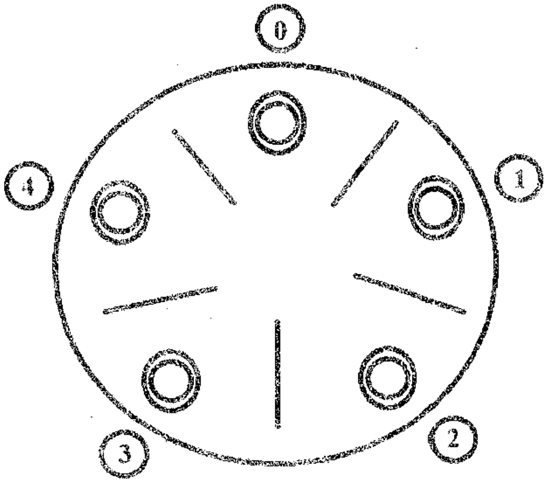
```
bool flag[2] = {false, false};
int turn;

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;
```

i) What are the sections corresponding to critical sections?

ii) What is the use of "while" conditions in both processes?

[4 Marks]

c) 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.

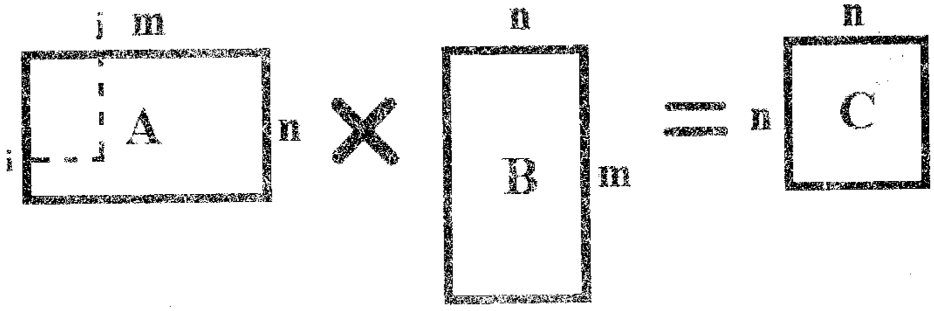


i) How can the philosophers end up with deadlock state?

ii) Explain how one philosopher has to starve by not having both forks to eat.

[4 Marks]

Q2 a) Write a parallel program using C/C++ and OpenMP to multiply two matrices.



Note: Fill the A and B matrices with the value i x j (i and j are row index and the column index)

[4 Marks]

b) Explain the OpenMP fork-join model using a diagram.

[2 Marks]

c) Explain the difference in execution of following two programs.

```
#pragma omp parallel
{
   #pragma omp for
   for(int i=0; i<N; i++)
   {

   }
}
```

```
#pragma omp parallel for
for(int i=0; i<N; i++)
{

}
```

[2 Marks]

d) What is the output of the following program?

```
#define NUM_THREADS     5
 void *PrintHello(void *threadid)
 {
    long tid;
    tid = (long)threadid;
    printf("Hello World! It's me, thread #%ld!\n", tid);
    pthread_exit(NULL);
 }

 int main (int argc, char *argv[])
 {
    pthread_t threads[NUM_THREADS];
    int rc;
    long t;
    for(t=0; t<NUM_THREADS; t++){
       printf("In main: creating thread %ld\n", t);
       rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
       if (rc){
          printf("ERROR; return code is %d\n", rc);
          exit(-1);
       }
    }
    pthread_exit(NULL);
 }
```

[2 Marks]

Q3 Consider the following program written using CUDA C

```
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
```

a) How many number of thread blocks are used in above CUDA program?

[1 Mark]

b) What is the maximum number you get for `threadIdx.x` ?

[1 Mark]

c) Modify the above program to work with thread block size of 16 x 16 to add matrices A[N][N] and B[N][N].

[5 Marks]

d) Consider the following CUDA program

```
size_t size = N * sizeof(float);
float* h_A = (float*)malloc(size);
float* d_A;
cudaMalloc(&d_A, size);
cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
```

i) Explain the function `cudaMalloc(&d_A, size)`.

ii) Explain the function `cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice)`.

[3 Marks]

Q4 Consider the following C# program

```
public static void ThreadMethod(object o)
{
    for (int i = 0; i < (int)o; i++)
    {
        Console.WriteLine("ThreadProc: {0}", i);
        Thread.Sleep(0);
    }
}
public static void Main()
{
    Thread t = new Thread(new
ParameterizedThreadStart(ThreadMethod));
    t.Start(5);
    t.Join();
}
```

a) What is the output of the above program?

[2 Marks]

b) What is the difference between foreground and background threads in C#?

[2 Marks]

c) In MapReduce framework, What are the operations applied to key-value pairs when those key-value pairs move from map function to the reduce function?

[2 Marks]

d) Following map function and the reduce function are of a word count program. Complete the TWO parameters of **`contex.write( , )`** function in the map function and the reduce function.

```
private final static IntWritable one = new IntWritable(1);
private Text word = new Text();

public void map(..)
StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
      word.set(itr.nextToken());
      context.write( , );
    }
}

public void reduce(Text key, Iterable<IntWritable> values,
        Context context) {
    int sum = 0;
    for (IntWritable val : values) {
      sum += val.get();
    }
    result.set(sum);
    context.write( , );
}
```

[4 Marks]

Q5 a) What is the difference between distributed memory computing and shared memory computing?

[2 Marks]

b) What is the information we can get from the MPI function `MPI_Comm_rank(MPI_COMM_WORLD, &world_rank)`?

[2 Marks]

c) What is the information we can get from the MPI function `MPI_Comm_size(MPI_COMM_WORLD, &world_size)`?

[2 Marks]

d) What is the type of the message sent with **MPI_Send()**, Is it synchronous or asynchronous?

[2 Marks]

d) Describe the output of the following MPI program using a diagram when `world_size=5`.

```
MPI_Init(NULL, NULL);
int world_rank;
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);

int token;
if (world_rank != 0) {
  MPI_Recv(&token, 1, MPI_INT, world_rank - 1, 0, MPI_COMM_WORLD,
             MPI_STATUS_IGNORE);
  printf("Process %d received token %d from process %d\n", world_rank,
          token, world_rank - 1);
} else {
    token = -1;
}
MPI_Send(&token, 1, MPI_INT, (world_rank + 1) % world_size, 0,
           MPI_COMM_WORLD);

if (world_rank == 0) {
    MPI_Recv(&token, 1, MPI_INT, world_size - 1, 0, MPI_COMM_WORLD,
             MPI_STATUS_IGNORE);
    printf("Process %d received token %d from process %d\n", world_rank,
             token, world_size - 1);
}
MPI_Finalize();
```

[2 Marks]