Student Name/Roll Number :

National University of Computer and Emerging Sciences

CS428 Concurrent & Distributed Systems

Spring 2021 Final Examination

Time Allowed: **3 Hours** Instructor: **Dr. Omar Usman Khan**

Q1 Briefly present at least two techniques that may be able to reduce the execution time of a software, without the need of increasing number of threads. (**4**)

Q2 What is Moore's Law? (**2**)

Q3 What is the difference between task parallel and data parallel models of concurrent programming. Present your answer with some simple supportive examples. (**4**)

Q4 Write a multi-threaded program using the POSIX threading library. The program should be able to create 10 threads. Each thread should simply display a *hello* message. The master thread should wait for all these threads to execute, and then display a message saying *all threads are finished*. (**4**)

Q5 What difference does it make if a gpu program is run as 1 block with 8 threads, or 8 blocks with 1 thread? (**3**)

Q6 Explain what a *warp* is, and why it is needed in GPU programs? (**4**)

Q7 What is the rationale behind usage of the symbol __global__ when declaring cuda kernel functions? (**3**)

Q8 What would be required at Position 1 and 2 of the code below to get the output: 11 0 22 0 33 0 44 0 55 0 66 0 77 0 88 0 (**4**)

```
__global__ void myHelloOnGPU(int *array) {
  // Position 1
}
int main() {
  int N = 16;
  int *cpu = (int*)malloc(sizeof(int)*N);
  int *gpu;
  cudaMalloc((void **)&gpu, sizeof(int)*N);
  // Position 2
  cudaMemcpy(cpu, gpu, sizeof(int)*N,
          cudaMemcpyDeviceToHost);
  for (int i = 0; i < N; i++) {
    printf("%d ", cpu[i]);
  }
}
```

for Pos 1
int c=1
for (int i=0; i<N; i++)
if(i%2==0)
array[i] = 11*c;
c=c+1
else
array[i] = 0;
Pos2 Init irray
for(int i=0; i<N; i++)
array[i]=i;
myHelloOnGPU <<<N,1>>>(cpu,gpu,N)

Q9 Write code using OpenMP for computing the dot product of two arrays A and B. A dot product is given as $D = \sum_{i=0}^{n} a_i b_i = a_0 b_0 + a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$ (**4**)

Q10 What is the difference between *taskwait* and *barrier* OpenMP High Level Synchronization directives? (**3**)

Q11 Explain how race conditions can be avoided in asynchronous point to point communications in MPI. (**4**)

Q12 Explain the differences between blocking and non-blocking communications of MPI. (**4**)

Q13 Design an MPI based communication system comprising of 4 processor ranks (0, 1, 2, 3), where rank 0 sends a point to point blocking message to rank 1, 2, and 3. (**4**)

Expected Attempt Time: 137 minutes
Total Paper Marks: 47
Paper Weightage: 60%

List of Signatures

```
int pthread_create(pthread_t *t, pthread_attr_t *a, void*(*function), void *arg);
int pthread_join(pthread_t t, void **retvalue);
void pthread_exit(void *retval);

int MPI_Test(MPI_Request *req, int *flag, MPI_Status *status)
int MPI_Probe(int source,  int tag, MPI_Comm comm, MPI_Status *status)
int MPI_Get_count(MPI_Status *status, MPI_Datatype datatype, int *count)
int MPI_Send(void *buffer, int count, int destination, int tag, MPI_DATATYPE datatype,
             MPI_Comm comm);
int MPI_Isend(void *buf, int count, MPI_Datatype datatype, int dest, int tag,
              MPI_Comm comm, MPI_Request *request)
int MPI_Recv(void *buffer,  int count,  MPI_DATATYPE datatype, int source, int tag,
             MPI_Comm comm, MPI_Status *status);
int MPI_Irecv(void *buf, int count, MPI_Datatype datatype, int src, int tag,
              MPI_Comm comm, MPI_Request *request)
int MPI_Wait(MPI_Request *req, MPI_Status *status)
int MPI_Waitall(int x, MPI_Request *req, MPI_Status *stat)
int MPI_Barrier(MPI_Comm comm)
int MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)
int MPI_Scatter(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf,
                int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
int MPI_Gather(void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf,
               int  recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype,
               MPI_Op op, int root, MPI_Comm comm)
int MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)
```