CS140. Exercise 3 MPI and Program Parallelization

Select the best choice for each question. For MPI code, there is no compilation error and the code syntax is correct.

1. Given the following program in which Process 0 sends and Process 1 receives, what is the printing result of printf().

```
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int buf[2]={1,2};
buf[0]=rank;
if (rank == 0) {
        MPI_Send( buf, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
 } else if (rank == 1) {
        MPI_Recv( buf, 2, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf( "%d, %d\n", buf[0],buf[1]);
}
```

(a) 1, 2 (b) 0, 2 (c) 2, 0 (d) 2, 2 (e) execution hangs
(f) None of above is correct

2. Given this MPI program segment, what is the printing result of printf() statement? Assume there are 3 processes in total.

```
int i, rank, buf[2];
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
buf[0]=buf[1]=rank;
if (rank== 0) {
        MPI_Send( buf, 2, MPI_INT, 2, 0, MPI_COMM_WORLD);
} else if (rank == 1) {
        MPI_Send( buf, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);
}else if (rank == 2) {
        for (i= 0; i<2; i++) {
          MPI_Recv(buf, 2, MPI_INT, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        }
        printf( "%d,%d\n", buf[0],buf[1]);
}
```

(a) 1, 0 (b) 0, 0 (c) 1, 1 (d) 2, 2 (e) Execution hangs
(f) None of above is correct

3. With 3 processes running, what is the printing output of the printf() statement?

```
Int rank,  buf[2];
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
buf[0]=buf[1]=rank;
MPI_Bcast (buf, 1, MPI_INT, 1, MPI_COMM_WORLD);
if( rank==0) {
        printf( "%d, %d\n", buf[0],buf[1]);
}
```

(a) 0, 0 (b) 1, 1 (c) 2, 2 (d) 1,0 (e) Execution hangs
(f) None of above is correct

4. With 3 processes running, what is the printing output of the printf() statement?

```
int rank, x=5, sum=1;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
If(rank!=0) {
        MPI_Reduce ( &rank, &sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
} else  {
        MPI_Reduce ( &sum, &x, 1,MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
}
if(rank==0) {
        printf( "%d %d\n", x, sum);
}
```

(a) 3 1 (b) 5 4 (c) 4 5 (d) 4 1
(e) This program hangs and nothing prints out.
(f) None of above is correct

5. With 3 processes running, what is the print output of the printf() statement?

```
int rank, sum[2], b[2], final[6];
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
b[0]=rank; b[1]=rank+1;
sum[0]=sum[1]=0;
MPI_Reduce ( b, sum, 2, MPI_INT, MPI_MAX, 1, MPI_COMM_WORLD);
MPI_Allgather(sum, 2, MPI_INT, final, 2, MPI_INT, MPI_COMM_WORLD);
if(rank==2) printf( "%d %d\n", final[0],final[1]);
```

(a) 0 0 (b) 0 1 (c) 3 6
(d) Execution hangs and nothing prints out.
(e) None of above is correct

6. Given the following sequential code,

S1: A=3+B+C;
S2: B= 3+ D;
S3: C= A+B;

Select a false statement from the following 5 choices on the dependence among S1, S2, and S3 following the above sequential program semantics.

(a) There is an anti-dependence from S1 to S3
(b) There is an anti-dependence from S1 to S2
(c) There is a true dependence from S1 to S3
(d) There is an anti-dependence from S3 to S1
(e) There is a true dependence from S2 to S3

7. Given the following code sequential program segment:

```
for(i=1; i<=100; i++) {
  for(j=1; j<=100; j++) {
    a[i][j]= a[i-1][j] +a[i][j+1]; //Call this statement as S_{i,j}
  }
}
```

Some student says the following anti or true dependence edges exist in the 2D iteration space of the above loop: (1) $S_{50, 50} \rightarrow S_{51, 50}$ (2) $S_{50, 51} \rightarrow S_{50, 50}$ (3) $S_{50, 50} \rightarrow S_{50, 51}$

Select a true statement below

(a) Only (1) and (2) exist
(b) Only (1) and (3) exist
(c) Only (2) and (3) exist
(d) All of 3 edges exist
(e) Only one of 3 edges exists.
(f) None of these 3 edges exists.

8. Given a correct sequential code with integer n and array a[][], is it legal to perform loop exchange for the following code segment?

```
 for(j=1; j<=n; j++) {
   for(i=1; i<=n; i++) {
        a[i][j]= a[i-1][j] +a[i][j+1];
  }
}
```

(a) Legal (b) not legal (c) Sometime illegal, depending on value of integer n.

(d) There is not enough information to judge

9. Perform loop interchange of the following program segment

```
for(i=2; i<=n; i++){
  for(j=i-1; j<=i+1; j++){
          a[i][j]= 1+a[i][j];
  }}
```

The new program is:

```
Lj=??;  Uj=??;  Li=??; Ui=??;
for(j=Lj; j<=Uj; j++){
  for(i=Li; i<=Ui; i++){
          a[i][j]= 1 +a[i][j];
  }}
```

Fill the lower and upper bound values for Loop j

(a) Lj= 2; Uj = n; (b) Lj= 1; Uj=n+1 (c) Lj= i-1; Uj=i+1
(d) None of above is correct

10. Continue from Question 9 and find bounds for Loop i.

(a) Li= 2; Ui = n; (b) Li= j-1; Ui=j+1 (c) Li= min(2,j-1), Ui=max(n,j+1)

(d) Li= max(2,j-1), Ui=min (n,j+1) (e) None of above is correct

11. The following pseudo code segment presented in the discussion section is SPMD-based code for parallel matrix vector multiplication.

```
int x[n], y[n], a[r][n];
me=mynode();
for i = 1 to n do
   if proc_map(i) == me, then do S_i:
      S_i :   y[i] = 0;
              for j = 1 to n do
                 y[i] = y[i] + a[local(i)][j] * x[j];
              endfor
endfor
```

There are p processes numbered 0 to p-1. Matrix a of size nxn is distributed evenly in p processes and each process holds r rows of this matrix in local array a[r][n]. This local array a[r][n] starts both row index and column index from 1. Vectors x and y also start from index position 1. Assume cycle mapping is used. Select a proper expression for function proc_map(i):

(a) floor ((i-1)/p) +1 (b) floor (i/p) (c) i mod p (d) (i mod p)+1 (e) (i-1) mod p

12. Continue from Question 11. Select a proper expression for function local(i):
(a) floor ((i-1)/p)+1 (b) floor (i/p) +1 (c) floor (i/p) (d) i mod p (e) (i mod p)+1
(f) (i-1) mod p+1

13. Given a task graph that manipulates some data items, we write MPI-based SPMD code that maps data and task computation to parallel processes. Choose a false statement from the following 4 choices
(a) Anti-dependence between two tasks does not need be enforced if these two tasks are executed in different MPI processes.
(b) True data dependence between two tasks does not need be enforced if two tasks are executed in the same MPI process.
(c) Owner computes rule defines where to map computation when data mapping to processes is determined first.
(d) Task scheduling for each process is specified by SPMD code segment.
(e) Cyclic mapping can outperform block mapping for better process load balancing.