Exam 1

Count numbers divided by 7 in array contain 150 elements from 1 to 150

Using Reduce:

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int arr[150],sen[50],rec =0 ;
for (int i = 1; i <= 150; i++)
      arr[i-1] = i;
MPI_Scatter(arr,50 , MPI_INT , sen , 50 , MPI_INT ,0, MPI_COMM_WORLD);
int count = 0;
for (int i = 0; i < 50; i++)
      if (sen[i] % 7 == 0)
            count++;
MPI_Reduce(&count , &rec ,1, MPI_INT, MPI_SUM , 0 , MPI_COMM_WORLD);
if (rank == 0)
      cout << rec<< endl ;
MPI_Finalize();
```

Using Gather:

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int arr[150], sen[50], rec[3];
for (int i = 1; i <= 150; i++)
      arr[i - 1] = i;
MPI_Scatter(arr, 50, MPI_INT, sen, 50, MPI_INT, 0, MPI_COMM_WORLD);
int count = 0;
for (int i = 0; i < 50; i++)
      if (sen[i] % 7 == 0)
            count++;
MPI_Gather(&count, 1, MPI_INT, rec, 1, MPI_INT, 0, MPI_COMM_WORLD);
int total = 0;
for (int i = 0; i < 3; i++)
      total += rec[i];
if (rank == 0)
      cout << total << endl;
MPI_Finalize();
```

Exam 2

Parallel array search

- write MPI parallel code to search for the target in an array containing numbers from 0 to 149.
- Print "rank id" if you find the target else print "1-"
- Test cases
    - Target =3 print ➔ 1
    - Target = 145 print ➔ 7 "If you have 8 ranks"
    - Target =200 print ➔ -1

Requirement

- Array declaration must be processed via master rank "any rank but must be one rank"
- All ranks must search for the target. "At least 3 working ranks"
- Result must be printed via master rank only.

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int x = 0;
int arr[150], arr2, arr3[50], sen[50], rec[4];
if (rank == 0) {
	cin >> x;

	for (int i = 1; i <= 150; i++)
		arr[i - 1] = i;

}
MPI_Bcast(&x , 1  , MPI_INT, 0 , MPI_COMM_WORLD);
MPI_Scatter(&arr, 50, MPI_INT, sen, 50, MPI_INT, 0, MPI_COMM_WORLD);
int r = -1 ;
for (int i = 0; i < 50; i++) {
	if (sen[i] == x) {

		r = rank;
		break;
	}
}
MPI_Gather(&r , 1 , MPI_INT,  rec , 1 , MPI_INT, 0, MPI_COMM_WORLD);
if (rank == 0) {
	for (int i = 0; i < 3; i++)
		if (rec[i] != -1)
			cout << rec[i] << endl;
		else
			cout << "---" << endl;
}
MPI_Finalize();
```

Exam 3

A Parallel program to iterate on an array containing 1000 element and add to each even number the value (2), subtract from each odd number the value (1) and sum “all” new numbers

Requirement

- Array declaration must be processed via master rank “any rank but must be one rank”
- All ranks must work and have its task. “At least 3 working ranks”
- Result must be printed via master rank only.

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD , &rank );
int arr[1000],sen[250], count =0 ;
if (rank == 0) {

        for (int i = 1; i <= 1000; i++) {
                arr[i-1] = i;
        }
}

MPI_Scatter(&arr , 250 , MPI_INT,sen , 250 , MPI_INT, 0, MPI_COMM_WORLD);
int total = 0 ;
for (int i = 0; i < 250; i++) {
        if (sen[i] % 2 == 0) {
                total += sen[i] + 2;
        }else
                total += sen[i] -1 ;
}
MPI_Reduce(&total ,&count , 1 ,MPI_INT , MPI_SUM , 0 , MPI_COMM_WORLD);
if (rank == 0) {

        cout << count << endl;
}
MPI_Finalize();
```

HPC Hands on

1 -> Hello world

```
int size;
int rank;
MPI_Init(NULL, NULL);
MPI_Comm_size(MPI_COMM_WORLD, &size);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
char pro[MPI_MAX_PROCESSOR_NAME];
int LEN;
MPI_Get_processor_name(pro, &LEN);
printf("Hello world from processor %s , rank is %d , size is %d \n", pro,
rank, size);
MPI_Finalize();
```

2 **->** Ping Pong

```
MPI_Init(NULL, NULL);
int  R, SENDE, RE, DATA;
MPI_Comm_rank(MPI_COMM_WORLD, &R);
if (R == 0)
{
        RE = 1;
        SENDE = 1;
        MPI_Send(&DATA, 1, MPI_INT, RE, 0, MPI_COMM_WORLD);
        MPI_Status S;
        MPI_Recv(&DATA, 1, MPI_INT, SENDE, 0, MPI_COMM_WORLD, &S);
        cout << "PING   ";
}
if (R == 1)
{
        RE = 0;
        SENDE = 0;
        MPI_Status S;
        MPI_Recv(&DATA, 1, MPI_INT, SENDE, 0, MPI_COMM_WORLD, &S);
        cout << "PONG   ";
        MPI_Send(&DATA, 1, MPI_INT, RE, 0, MPI_COMM_WORLD);
}
MPI_Finalize();
```

3 **->** Assume you have **n** nodes each one sends message to next on ring form
For example of n=4

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int data, sendr, rec;
switch (rank)
{
case 0:
        sendr = 1;
        rec = 3;
        MPI_Send(&data, 1, MPI_INT, sendr, 0, MPI_COMM_WORLD);
        cout << "hi from processor 0 " << endl;
        MPI_Status S;
        MPI_Recv(&data, 1, MPI_INT, rec, 0, MPI_COMM_WORLD, &S);
        break;
case 1:
        sendr = 2;
        rec = 0;
        MPI_Status o;
        MPI_Recv(&data, 1, MPI_INT, rec, 0, MPI_COMM_WORLD, &o);
        cout << "hi from processor 1 " << endl;
        MPI_Send(&data, 1, MPI_INT, sendr, 0, MPI_COMM_WORLD);
        break;
case 2:
        sendr = 3;
        rec = 1;
        MPI_Status p;
        MPI_Recv(&data, 1, MPI_INT, rec, 0, MPI_COMM_WORLD, &p);
        cout << "hi from processor 2 " << endl;
        MPI_Send(&data, 1, MPI_INT, sendr, 0, MPI_COMM_WORLD);
        break;
case 3:
        sendr = 0;
        rec = 2;
        MPI_Status x;
        MPI_Recv(&data, 1, MPI_INT, rec, 0, MPI_COMM_WORLD, &x);
        cout << "hi from processor 3 " << endl;
        MPI_Send(&data, 1, MPI_INT, sendr, 0, MPI_COMM_WORLD);
        break;
default:
        break;
}
MPI_Finalize();
```

4 -> send array

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int data[15], rec[15], sen;
for (int i = 0; i < 15; i++) {
        data[i] = i + 1;
}

if (rank == 0) {
        sen = 1;
        MPI_Send(data, 15, MPI_INT, 1, 0, MPI_COMM_WORLD);

}
if (rank == 1) {

        MPI_Status s;
        MPI_Recv(rec, 15, MPI_INT, 0, 0, MPI_COMM_WORLD, &s);
        cout << "Array receive from processor 0" << endl;
        for (int i = 0; i < 15; i++) {
                cout << rec[i] << endl;
        }

}
MPI_Finalize();
```

5 -> Array sum

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int data[15], rec[15], sen, sum = 0;
for (int i = 0; i < 15; i++) {
        data[i] = i + 1;
}

if (rank == 0) {
        sen = 1;
        int total = 0;
        for (int i = 0; i < 5; i++) {
                total += data[i];
        }
        MPI_Send(&data[5], 5, MPI_INT, 1, 0, MPI_COMM_WORLD);
        MPI_Send(&data[10], 5, MPI_INT, 2, 0, MPI_COMM_WORLD);
        MPI_Status s;
        MPI_Recv(&sen, 1, MPI_INT, 1, 0, MPI_COMM_WORLD, &s);
        total += sen;

        MPI_Recv(&sen, 1, MPI_INT, 2, 0, MPI_COMM_WORLD, &s);
        total += sen;
        cout << total << endl;
```

```
}
if (rank == 1 || rank == 2) {

        MPI_Status s;
        MPI_Recv(rec, 5, MPI_INT, 0, 0, MPI_COMM_WORLD, &s);
        cout << "Array receive from processor 0" << endl;
        sum = 0;
        for (int i = 0; i < 5; i++) {
                sum += rec[i];
        }
        MPI_Send(&sum, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
        MPI_Finalize();
```

6 -> Array sum using scatter and gather

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int data[15], rec[15], sen, sum = 0;
for (int i = 0; i < 15; i++) {
        data[i] = i + 1;
}


MPI_Scatter(data, 5, MPI_INT, rec, 5, MPI_INT, 0, MPI_COMM_WORLD);
sum = 0;
for (int i = 0; i < 5; i++) {
        sum += rec[i];
}
MPI_Gather(&sum, 1, MPI_INT, &rec, 1, MPI_INT, 0, MPI_COMM_WORLD);


if (rank == 0) {
        sen = 1;
        int total = 0;

        for (int i = 0; i < 3; i++) {
                total += rec[i];
        }

        cout << total << endl;
}
MPI_Finalize();
```

7 -> Dot Product

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int a1[100], a2[100], rec[25], rec2[25];
for (int i = 0; i < 100; i++) {
        a1[i] = 1;
        a2[i] = 2;
}
MPI_Scatter(a1, 25, MPI_INT, rec, 25, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Scatter(a2, 25, MPI_INT, rec2, 25, MPI_INT, 0, MPI_COMM_WORLD);
int dot = 0;
for (int i = 0; i < 25; i++) {
        dot += rec[i] * rec2[i];
}
MPI_Gather(&dot, 1, MPI_INT, rec, 1, MPI_INT, 0, MPI_COMM_WORLD);
if (rank == 0) {
        int total = 0;
        for (int i = 0; i < 4; i++) {
                total += rec[i];
        }
        cout << " dot product = " << total << endl;
}
MPI_Finalize();
```

8 -> Dot Product using Reduce

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int a1[100], a2[100], rec1[25], rec2[25], dot = 0, total = 0;
for (int i = 0; i < 100; i++) {
        a1[i] = 1;
        a2[i] = 2;
}
MPI_Scatter(a1, 25, MPI_INT, rec1, 25, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Scatter(a2, 25, MPI_INT, rec2, 25, MPI_INT, 0, MPI_COMM_WORLD);
for (int i = 0; i < 25; i++) {
        dot += rec1[i] * rec2[i];
}
MPI_Reduce(&dot, &total, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
if (rank == 0)
        cout << total << endl;
MPI_Finalize();
```

9 ->

$$\pi = \int_0^1 \frac{dx}{1+x^2} \approx \frac{4}{N}\sum_{i=1}^{N} \frac{1}{1+\left(\frac{i-\frac{1}{2}}{N}\right)^2}$$

```
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
int rev[25];
float result = 0, rec;
for (float i = 25 *rank + 1; i <= 25 * rank + 25; i++) {
        rec = ((i - 0.5) * (i - 0.5)) / 10000;
        result += (1 / (1 + rec));
}
result = result * 4 / 100;
MPI_Reduce(&result, &rec, 1, MPI_FLOAT, MPI_SUM, 0, MPI_COMM_WORLD);

if (rank == 0)
        cout << rec << endl;
MPI_Finalize();
```