

Register Number : 20BCE1294

Name : Arnab Mondal

Question 1:

Code:

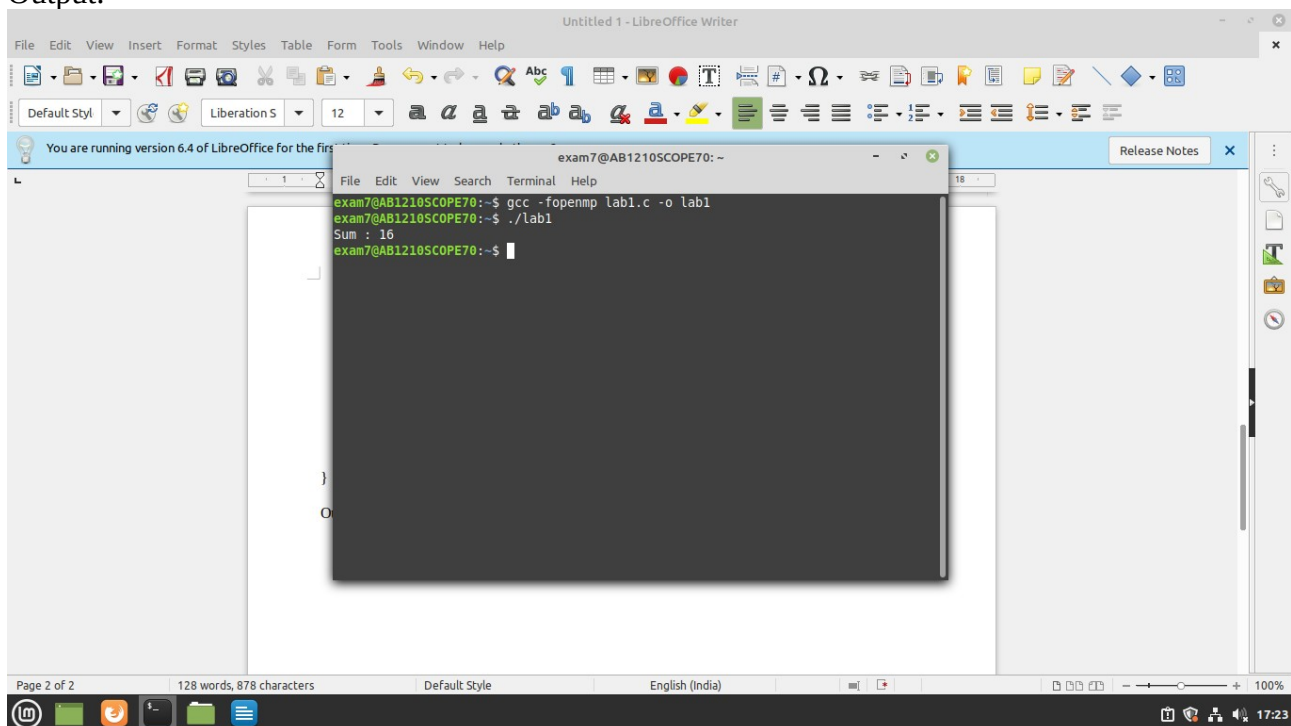
```
#include <omp.h>
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
#include <sys/time.h>
#include <stdio.h>
#define n1 6
int main()
{
    int m=2,n=1,p=2,i;
    for(i=1;;i++)
    {
        if(i%2==0)
        {
            //printf("%d, ",m);
            m+=m;
        }
        else if(i%2!=0)
        {
            //printf("%d, ", n);
            n+=p;
            p++;
        }
        if(n>n1 && m>n1)
            break;
    }
    //printf("\n%d",i);
    int k=i;
    int arr[i];
    m=2,n=1,p=2;
    omp_set_num_threads(5);
    #pragma omp parallel for private(i) shared(arr) ordered
    for(i=1;i<=k;i++)
    {
        if(i%2==0)
        {
            //printf("%d, ",m);
            arr[i-1] = m;
            m+=m;
        }
        else if(i%2!=0)
        {
            //printf("%d, ", n);
            arr[i-1] = n;
            n+=p;
            p++;
        }
    }
}
```

```

        //if(n>n1 && m>n1)
        //break;
    }
    int sum = 0;
    #pragma omp parallel for private(i) shared(arr) reduction(+: sum) ordered
    for (i = 0; i < n1; ++i) {
        sum += arr[i];
    }
    printf("Sum : %d\n",sum);
    return 0;
}

```

Output:



Question 2 :

Code:

```

#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#define n 5
int main(int argc, char * argv[]){
    int j=0,count=0;
    for(j=1;j<=n;j++)
    {
        count++;
        j++;
    }
}

```

```

int a[100],a2[100];
int elements_per_process,elements_left, n_elements_recieved;
int np, pid;
int m=0;
for(j=1;j<=n;j++)
{
    a[m] = j;
    m++;
    j++;
}
MPI_Status status;
MPI_Request request = MPI_REQUEST_NULL;
MPI_Init( & argc, & argv);
MPI_Comm_size(MPI_COMM_WORLD, & np);
MPI_Comm_rank(MPI_COMM_WORLD, & pid);
if (pid == 0) {
    int index, i;
    elements_per_process = count / np;
    if (np > 1) {
        for (i = 1; i < np - 1; i++) {
            index = i * elements_per_process;
            MPI_Send( & elements_per_process,1, MPI_INT, i,
0,MPI_COMM_WORLD);
            MPI_Send( & a[index],elements_per_process,MPI_INT, i,
0,MPI_COMM_WORLD);
        }
        index = i * elements_per_process;
        int elements_left = count - index;
        MPI_Send( & elements_left,1, MPI_INT,i, 0,MPI_COMM_WORLD);
        MPI_Send( & a[index],elements_left,MPI_INT, i, 0,MPI_COMM_WORLD);
    }
    int sum = 0;
    for (i = 0; i < elements_per_process; i++)
        sum += a[i];
    int tmp;
    for (i = 1; i < np; i++)
    {
        MPI_Recv(&tmp, 1, MPI_INT, MPI_ANY_SOURCE, 0,
MPI_COMM_WORLD, &status);
        int sender = status.MPI_SOURCE;
        sum += tmp;
    }
    printf("Sum of array is : %d\n", sum);
}
else
{
    MPI_Recv(&n_elements_recieved,1, MPI_INT, 0, 0,MPI_COMM_WORLD,
&status);
    MPI_Recv(&a2, n_elements_recieved,MPI_INT, 0, 0,MPI_COMM_WORLD,
&status);
    int partial_sum = 0;
    for (int i = 0; i < n_elements_recieved; i++)

```

```

        partial_sum += (a2[i]*a2[i]);
        MPI_Send(&partial_sum, 1, MPI_INT,0, 0, MPI_COMM_WORLD);
    }
    MPI_Finalize();
    return 0;
}

```

Output:

The screenshot shows a code editor on the left and a terminal on the right. The code editor displays the source file `lab4.c` with the following C code:

```

index = i * elements_per_process;
MPI_Send( &elements_per_process,1, MPI_INT, i, 0,MPI_COMM_WORLD);
MPI_Send( &a[index],elements_per_process,MPI_INT, i, 0,MPI_COMM_WORLD);
}
index = i * elements_per_process;
int elements_left = count - index;
MPI_Send( &elements_left,1,MPI_INT,i, 0,MPI_COMM_WORLD);
MPI_Send( &a[index],elements_left,MPI_INT, i, 0,MPI_COMM_WORLD);
}
int sum = 0;
for (i = 0; i < elements_per_process; i++)
    sum += a[i];
int tmp;
for (i = 1; i < np; i++)
{
    MPI_Recv(&tmp, 1, MPI_INT, MPI_ANY_SOURCE, 0, MPI_COMM_WORLD, &status);
    int sender = status.MPI_SOURCE;
    sum += tmp;
}
printf("Sum of array is : %d\n", sum);
}
else
{
    MPI_Recv(&n_elements_recieved,1, MPI_INT, 0, 0,MPI_COMM_WORLD, &status);
    MPI_Recv(&a2, n_elements_recieved,MPI_INT, 0, 0,MPI_COMM_WORLD, &status);
    int partial_sum = 0;
    for (int i = 0; i < n_elements_recieved; i++)
        partial_sum += (a2[i]*a2[i]);
    MPI_Send(&partial_sum, 1, MPI_INT,0, 0, MPI_COMM_WORLD);
}
MPI_Finalize();
return 0;
}

```

The terminal on the right shows the execution of the program:

```

exam7@AB1210SCOPE70:~$ mpicc lab4.c -o lab4
exam7@AB1210SCOPE70:~$ mpirun -np 2 ./lab4
Sum of array is : 35
exam7@AB1210SCOPE70:~$

```

The terminal output confirms that the program executed successfully and produced the expected output: "Sum of array is : 35".