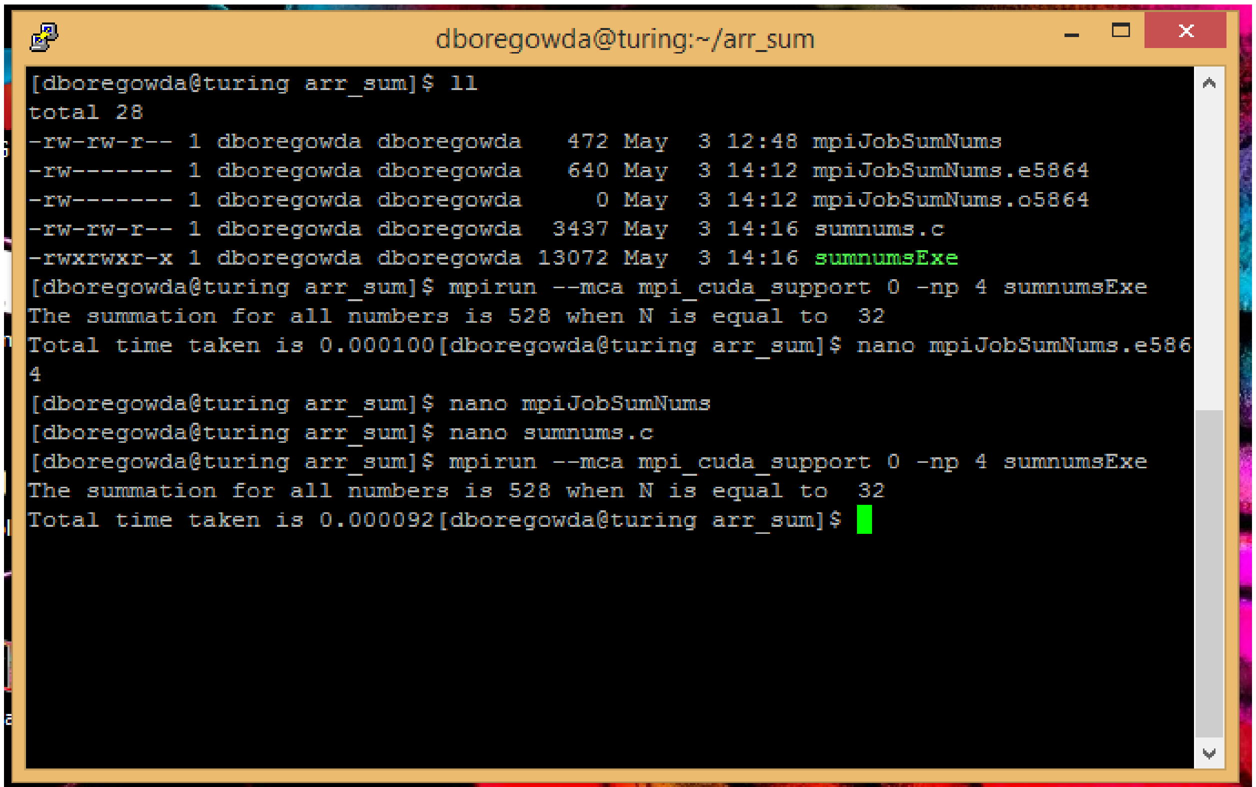
```
//NAMES: DivyaShree H B

//required header files
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

//*********************************************************************
//  Function : total_sum()
//          Parameters: int * array, int numItems
//  It calculates sum by adding items in the array
//*********************************************************************
int total_sum(int * array, int numItems){
          int tSum=0,i;
          for(i=0;i<numItems;i++){
                    tSum=tSum+array[i];
          }
          return tSum;
}

void main(){

          int my_rank, comm_sz, my_id;  // MPI variables
          int number = 32;           //number to which summation is calculated
          double start, finish;      // variables for calculating time
          int i,j,k,l,m,p;
          int count;                 //number of values assigned to each processor
          //                  //other required variables, such as loop control vars

          int recv_sum;              //sum received from slave processors
          int master_sum = 0;        //sum calculated by master process
          int partial_sum = 0;       //sum calculated by slave processors
          int final_sum;             //store final sum
          //Int temp_sum=0;

          //3 required MPI functions
          MPI_Init(NULL, NULL);
          MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);
          MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

          //determine how many numbers should be assigned to each process
          count = (number/comm_sz);

   //master process is 0
```

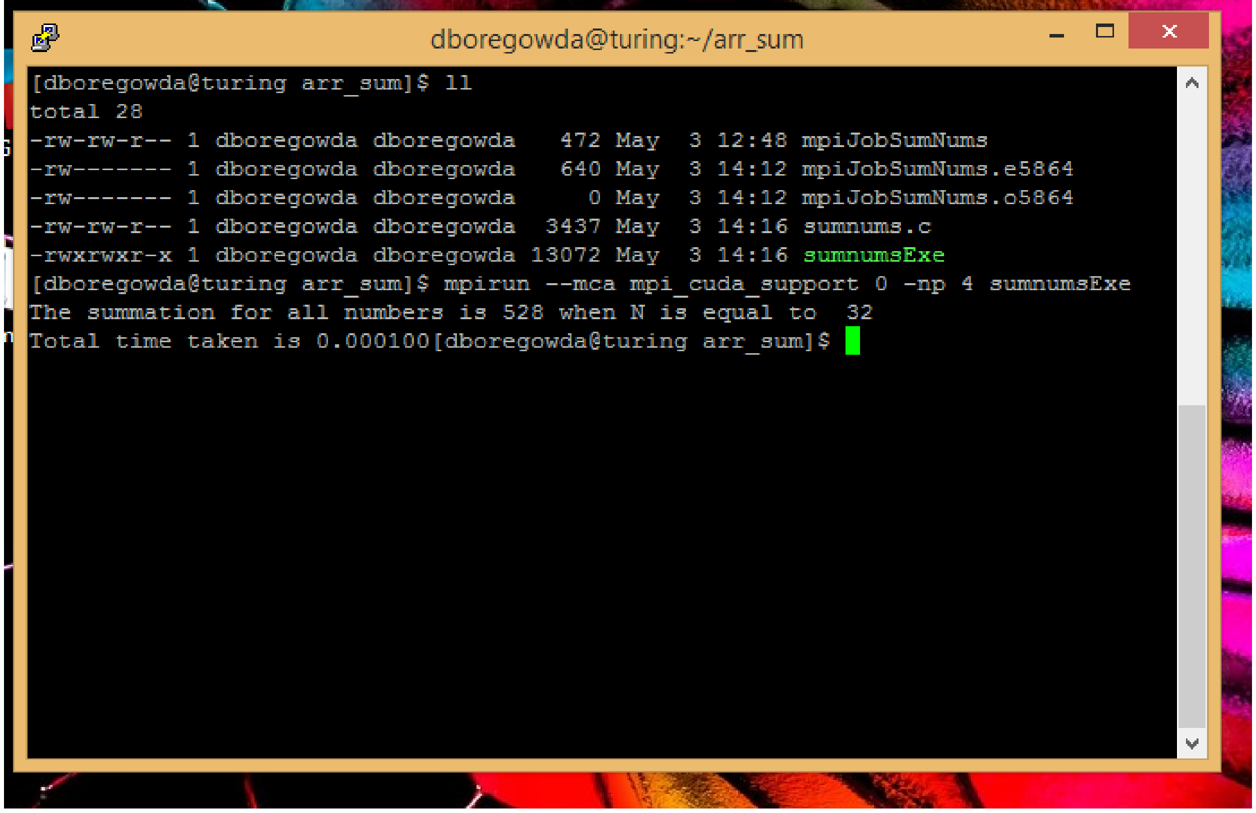
```
        if (my_rank == 0)  {

          //create array to store numbers
          int * array = (int *)malloc(sizeof(int)* number);

          //store values in the array from 1 to declared number
          for(i=0;i<number;i++){
                array[i]=i+1;
        }

    //create array to store partial sum calculated by each process
          int * summation = (int *)malloc(sizeof(int)* comm_sz);

    my_id = 1;

          start = MPI_Wtime();  //calculate time from this point

          //Loop to send values to each process - this is the hard part
          for (k=count; k< number; k+=count){
                 MPI_Send(array+k,count, MPI_INT, my_id, 0, MPI_COMM_WORLD);
                 //increment my_id
                 my_id++;
          }

          // Calculate sum for the values assigned to master process from the array
                for(m=0;m<count;m++){
                        master_sum=master_sum+array[m];
                }
        // store sum of master process in correct index of summation array
                summation[0]=master_sum;
          // Loop to receive sums from each slave process
    for (j=1;j<comm_sz;j++){
                 MPI_Recv(&recv_sum, 1, MPI_INT, j, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
                 //store in correct index of summation array
                 summation[j]=recv_sum;

          }

          //Call  total_sum to calculate sum of partial sums in summation
          final_sum=total_sum(summation,comm_sz);

          finish = MPI_Wtime();  // stop time at this point
```

```
            printf("The summation for all numbers is %llu when N is equal to  %lld  \n", final_sum,
number);
            printf("Total time taken is %f", finish - start);
          }

          //slave processes receive values sent by master process
          else {


            //declare recv_data array to store "count" ints sent by master process to this process
          int recv_data[count];

            //Receive the data from the master process

MPI_Recv(recv_data,count,MPI_INT,0,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);

            //Call total_sum to calculate the partial sum

       for(p=0;p<count;p++)
       {
                    partial_sum =partial_sum + *(recv_data+p);
       }


            //Send the sum of each slave process to master process
                    MPI_Send(&partial_sum ,1,MPI_INT,0,0,MPI_COMM_WORLD);

          }

          MPI_Finalize(); //MPI final
}
```



dboregowda@turing:~/arr_sum

```
[dboregowda@turing arr_sum]$ ll
total 28
-rw-rw-r-- 1 dboregowda dboregowda   472 May  3 12:48 mpiJobSumNums
-rw------- 1 dboregowda dboregowda   640 May  3 14:12 mpiJobSumNums.e5864
-rw------- 1 dboregowda dboregowda     0 May  3 14:12 mpiJobSumNums.o5864
-rw-rw-r-- 1 dboregowda dboregowda  3437 May  3 14:16 sumnums.c
-rwxrwxr-x 1 dboregowda dboregowda 13072 May  3 14:16 sumnumsExe
[dboregowda@turing arr_sum]$ mpirun --mca mpi_cuda_support 0 -np 4 sumnumsExe
The summation for all numbers is 528 when N is equal to  32
Total time taken is 0.000100[dboregowda@turing arr_sum]$ nano mpiJobSumNums.e586
4
[dboregowda@turing arr_sum]$ nano mpiJobSumNums
[dboregowda@turing arr_sum]$ nano sumnums.c
[dboregowda@turing arr_sum]$ mpirun --mca mpi_cuda_support 0 -np 4 sumnumsExe
The summation for all numbers is 528 when N is equal to  32
Total time taken is 0.000092[dboregowda@turing arr_sum]$
```



dboregowda@turing:~/arr_sum

```
[dboregowda@turing arr_sum]$ ll
total 28
-rw-rw-r-- 1 dboregowda dboregowda   472 May  3 12:48 mpiJobSumNums
-rw------- 1 dboregowda dboregowda   640 May  3 14:12 mpiJobSumNums.e5864
-rw------- 1 dboregowda dboregowda     0 May  3 14:12 mpiJobSumNums.o5864
-rw-rw-r-- 1 dboregowda dboregowda  3437 May  3 14:16 sumnums.c
-rwxrwxr-x 1 dboregowda dboregowda 13072 May  3 14:16 sumnumsExe
[dboregowda@turing arr_sum]$ mpirun --mca mpi_cuda_support 0 -np 4 sumnumsExe
The summation for all numbers is 528 when N is equal to  32
Total time taken is 0.000100[dboregowda@turing arr_sum]$
```