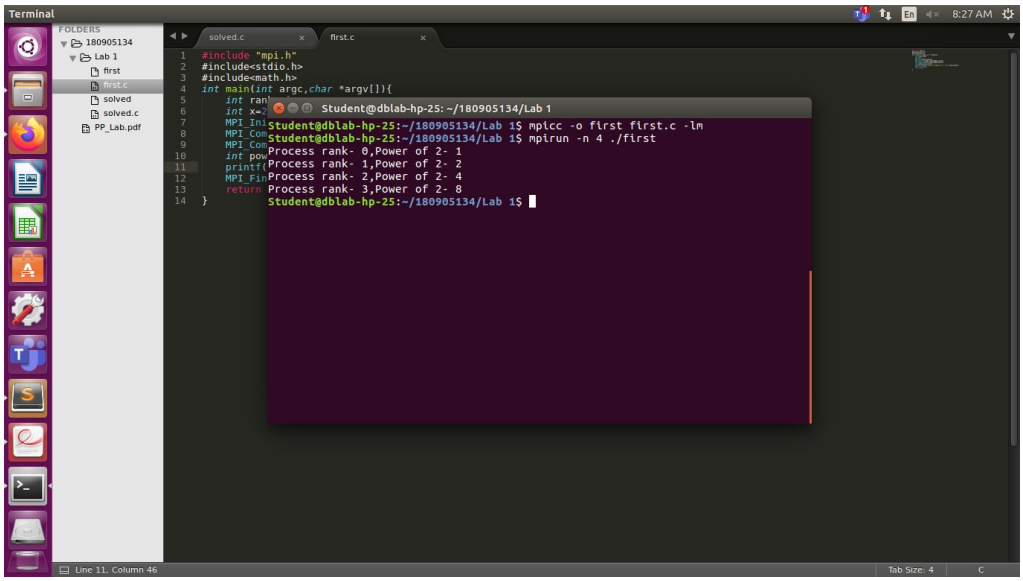
SANSKAAR PATNI
180905134 CSE C-23
LAB 1

1. CODE

```
#include "mpi.h"
#include<stdio.h>
#include<math.h>
int main(int argc,char *argv[]){
	int rank,size;
	int x =2;
	MPI_Init(&argc,&argv);
	MPI_Comm_rank(MPI_COMM_WORLD,&rank);
	MPI_Comm_size(MPI_COMM_WORLD,&size);
	int power=pow(x,rank);
	printf("Process rank- %d,Power of 2- %d\n",rank,power);
	MPI_Finalize();
	return 0;
}
```

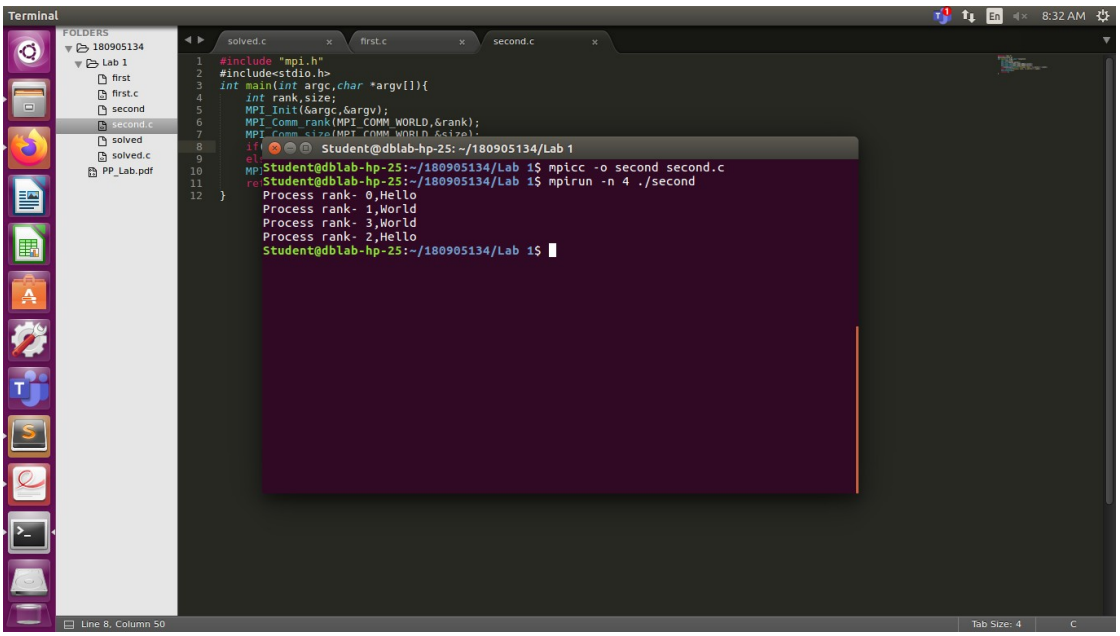
OUTPUT:



2.CODE

```
#include "mpi.h"
#include<stdio.h>
int main(int argc,char *argv[]){
	int rank,size;
	MPI_Init(&argc,&argv);
	MPI_Comm_rank(MPI_COMM_WORLD,&rank);
	MPI_Comm_size(MPI_COMM_WORLD,&size);
	if(rank%2==0)printf("Process rank- %d,Hello\n",rank);
	else printf("Process rank- %d,World\n",rank);
	MPI_Finalize();
	return 0;
}
```

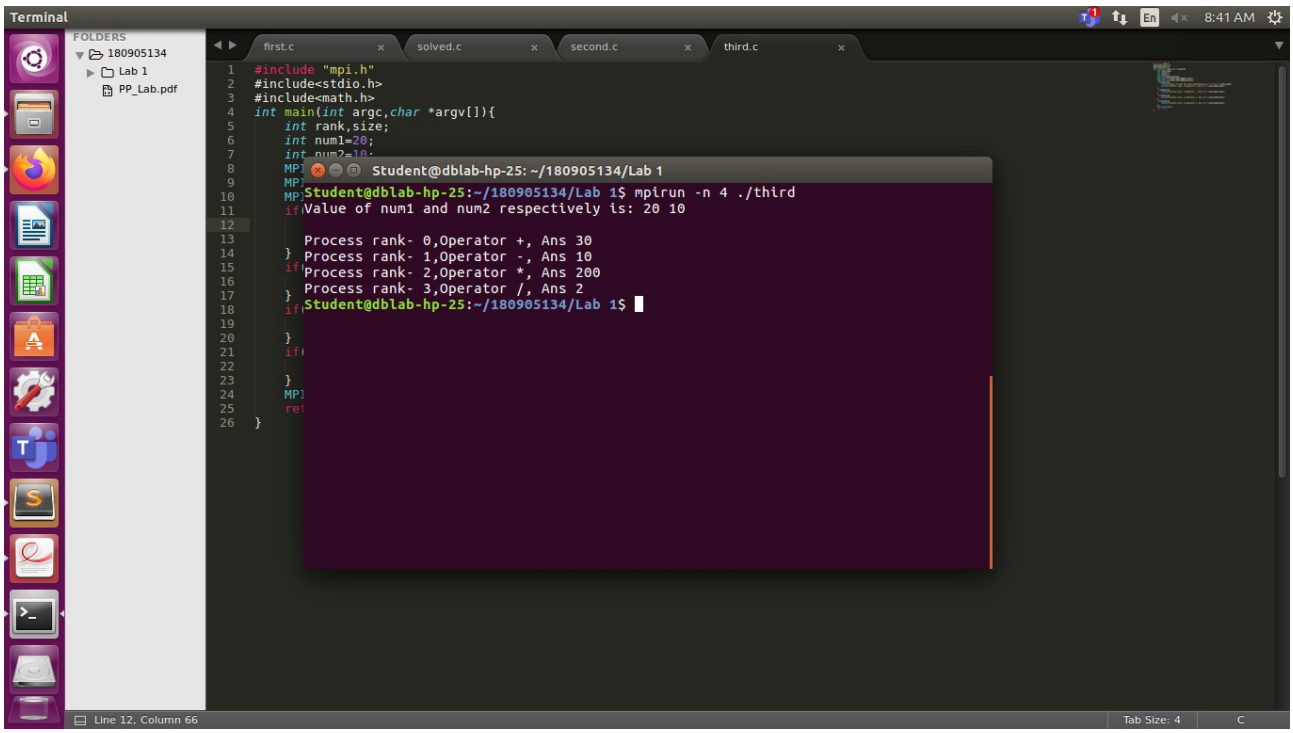
OUTPUT:



3.CODE

```
#include "mpi.h"
#include<stdio.h>
#include<math.h>
int main(int argc,char *argv[]){
        int rank,size;
        int num1=20;
        int num2=10;
        MPI_Init(&argc,&argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        if(rank==0){
                printf("Value of num1 and num2 respectively is: %d %d\n\n",num1,num2);
                printf("Process rank- %d,Operator +, Ans %d\n",rank,num1+num2);
        }
        if(rank==1){
                printf("Process rank- %d,Operator -, Ans %d\n",rank,num1-num2);
        }
        if(rank==2){
                printf("Process rank- %d,Operator *, Ans %d\n",rank,num1*num2);
        }
        if(rank==3){
                printf("Process rank- %d,Operator /, Ans %d\n",rank,num1/num2);
        }
        MPI_Finalize();
        return 0;
}
```

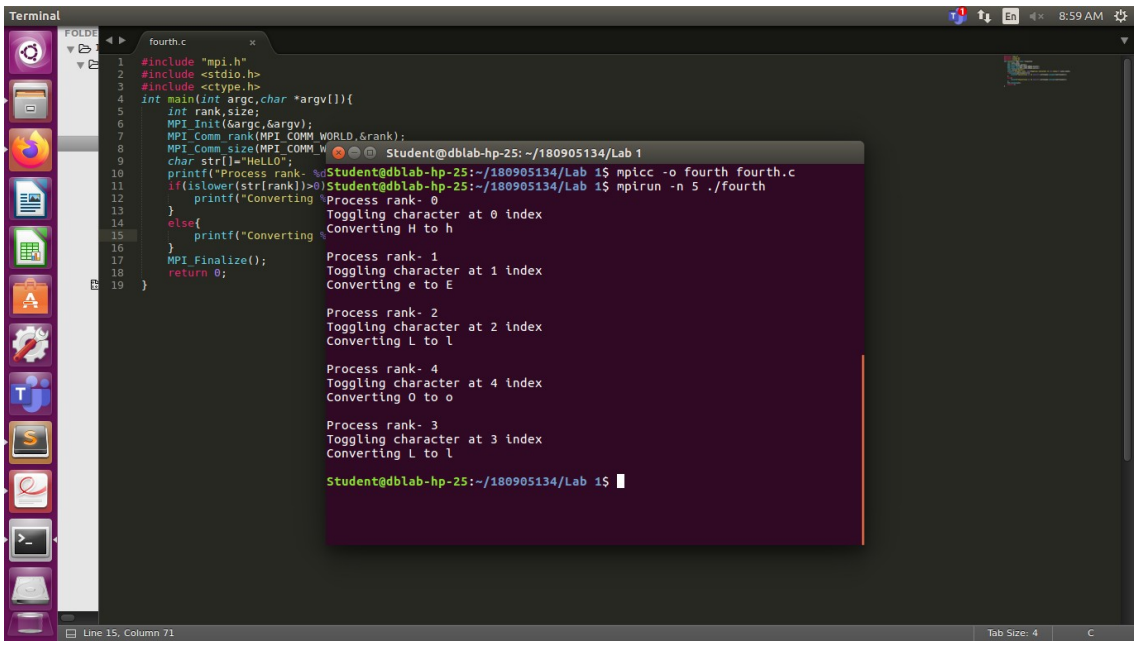
OUTPUT:



4.CODE

```
#include "mpi.h"
#include <stdio.h>
#include <ctype.h>
int main(int argc,char *argv[]){
        int rank,size;
        MPI_Init(&argc,&argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        char str[]="HeLLO";
        printf("Process rank- %d\nToggling character at %d index\n",rank,rank);
        if(islower(str[rank])>0){
                printf("Converting %c to %c\n\n",str[rank],toupper(str[rank]));
        }
        else{
                printf("Converting %c to %c\n\n",str[rank],tolower(str[rank]));
        }
        MPI_Finalize();
        return 0;
}
```

OUTPUT:

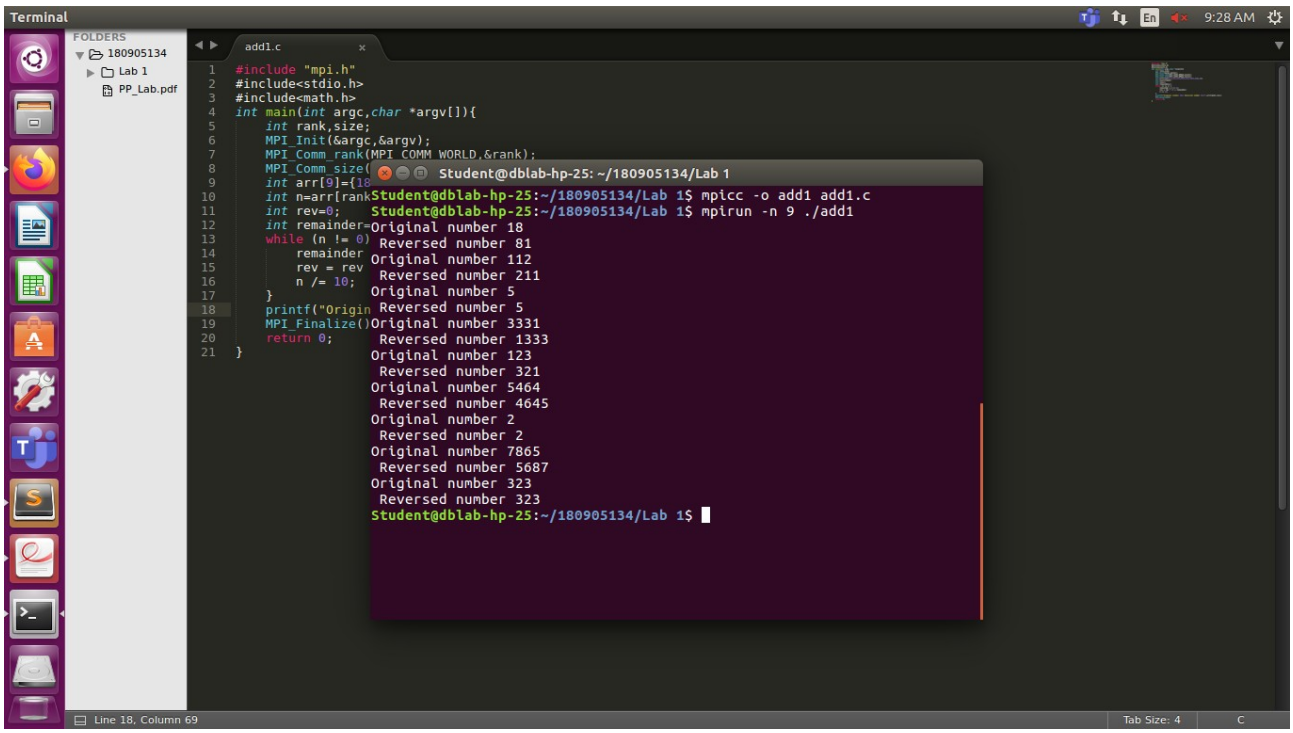


ADDITIONAL EXERCISES:

1.CODE

```
#include "mpi.h"
#include<stdio.h>
#include<math.h>
int main(int argc,char *argv[]){
        int rank,size;
        MPI_Init(&argc,&argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        int arr[9]={18,2,123,323,3331,112,7865,5464,5};
        int n=arr[rank];
        int rev=0;
        int remainder=0;
        while (n != 0) {
    remainder = n % 10;
    rev = rev * 10 + remainder;
    n /= 10;
  }
        printf("Original number %d\n Reversed number %d\n",arr[rank],rev);
        MPI_Finalize();
        return 0;
}
```

OUTPUT:



2.CODE

```
#include "mpi.h"
#include<stdio.h>
#include<math.h>
int main(int argc,char *argv[]){
        int rank,size;
        MPI_Init(&argc,&argv);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        if(rank==0){
                for (int i=2; i<50; i++)
                {
                        for (int j=2; j<=i; j++)
                        {
                        if (i == j)
                        printf("%d\n",i);
                        else if (i%j == 0)
                        break;
                        }
                }
        }else{
                for (int i=50; i<100; i++)
                {
                        for (int j=2; j<=i; j++)
                        {
                        if (i == j)
                        printf("%d\n",i);
                        else if (i%j == 0)
                        break;
                        }
                }
        }
```

```
        MPI_Finalize();
        return 0;
}
```

OUTPUT: