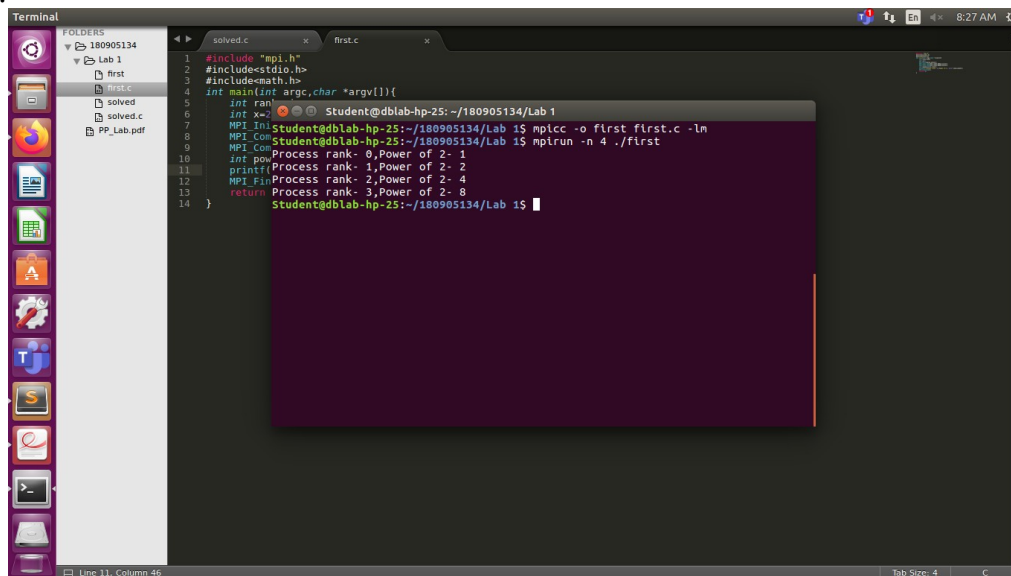


SANSKAAR PATNI
180905134 CSE C-23
LAB 1

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:

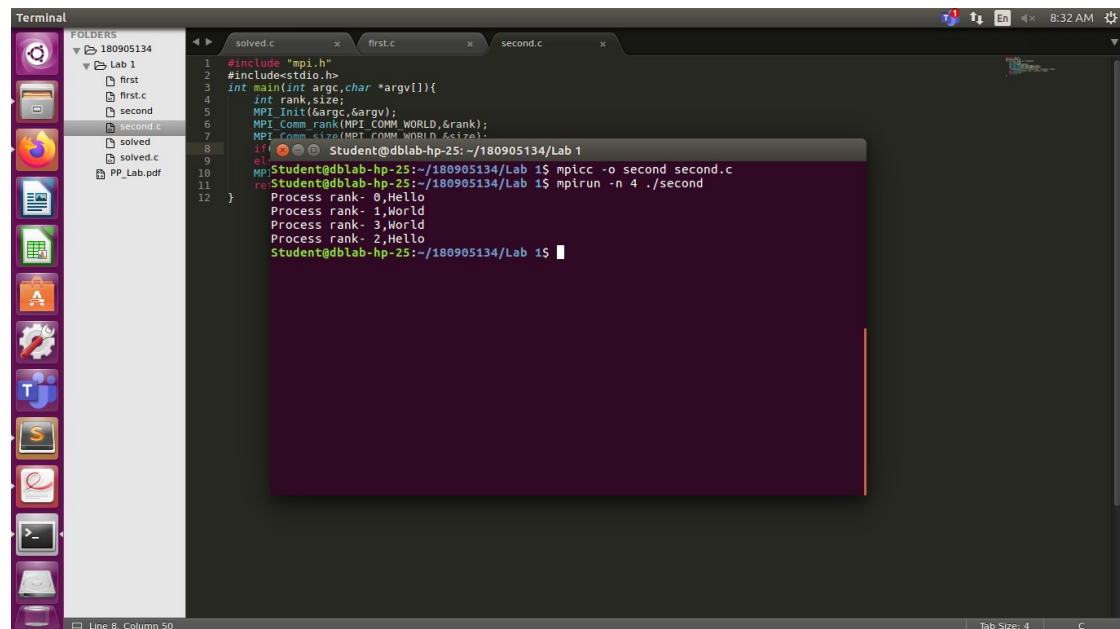


```
Student@dblab-hp-25i:~/180905134/Lab 1
Student@dblab-hp-25i:~/180905134/Lab 1$ mpicc -o first first.c -ln
Student@dblab-hp-25i:~/180905134/Lab 1$ mpirun -n 4 ./first
Process rank- 0, Power of 2- 1
Process rank- 1, Power of 2- 2
Process rank- 2, Power of 2- 4
Process rank- 3, Power of 2- 8
Student@dblab-hp-25i:~/180905134/Lab 1$
```

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:



```
1 #include "mpi.h"
2 #include<stdio.h>
3 int main(int argc,char *argv[]){
4     int rank,size;
5     MPI_Init(&argc,&argv);
6     MPI_Comm_rank(MPI_COMM_WORLD,&rank);
7     MPI_Comm_size(MPI_COMM_WORLD,&size);
8     if(rank==0){
9         printf("Value of num1 and num2 respectively is: %d %d\n\n",num1,num2);
10        printf("Process rank- %d,Operator +, Ans %d\n",rank,num1+num2);
11    }
12    if(rank==1){
13        printf("Process rank- %d,Operator -, Ans %d\n",rank,num1-num2);
14    }
15    if(rank==2){
16        printf("Process rank- %d,Operator *, Ans %d\n",rank,num1*num2);
17    }
18    if(rank==3){
19        printf("Process rank- %d,Operator /, Ans %d\n",rank,num1/num2);
20    }
21    MPI_Finalize();
22    return 0;
23 }
```

```
Student@dblab-hp-25:~/180905134/Lab 1
mpicc -o second second.c
mpirun -n 4 ./second
Process rank- 0,Hello
Process rank- 1,World
Process rank- 2,World
Process rank- 2,Hello
Student@dblab-hp-25:~/180905134/Lab 1 $
```

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;
}
```

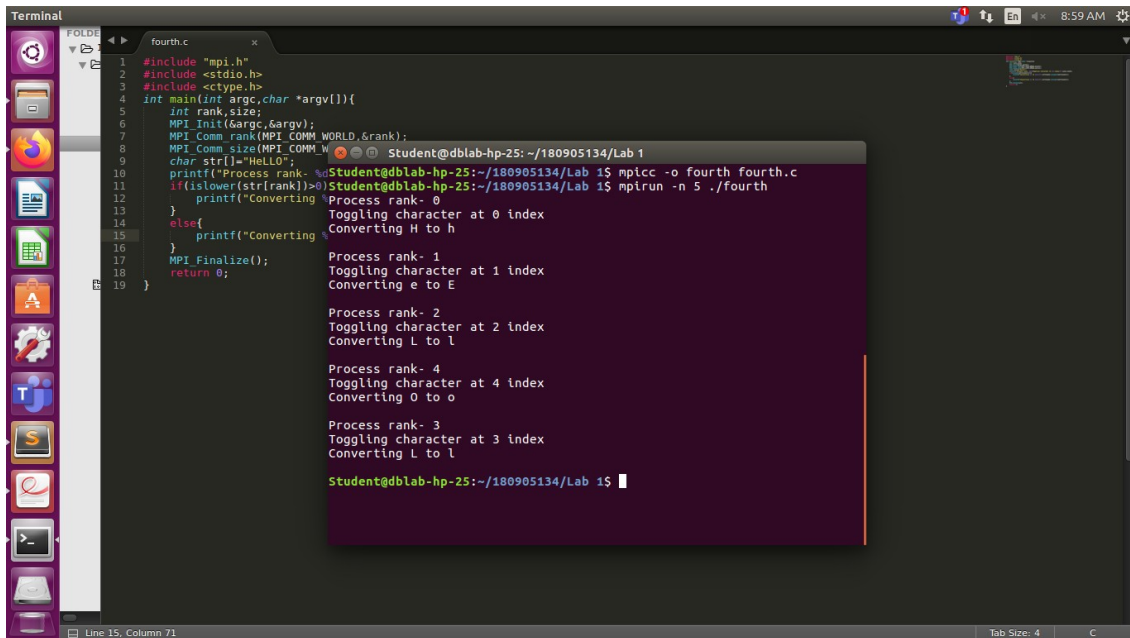
OUTPUT:

```
1 #include "mpi.h"
2 #include<stdio.h>
3 #include<math.h>
4 int main(int argc,char *argv[]){
5     int rank,size;
6     int num1=20;
7     int num2=10;
8     MPI_Student@dblab-hp-25: ~/180905134/Lab 1
9     MPI_Student@dblab-hp-25:~/180905134/Lab 1$ mpirun -n 4 ./third
10    Value of num1 and num2 respectively is: 20 10
11
12    Process rank- 0,Operator +, Ans 30
13    } Process rank- 1,Operator -, Ans 10
14    } Process rank- 2,Operator *, Ans 200
15    } Process rank- 3,Operator /, Ans 2
16    }
17    MPI_Student@dblab-hp-25:~/180905134/Lab 1$
```

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:



```
1 #include "mpi.h"
2 #include <stdio.h>
3 #include <ctype.h>
4 int main(int argc, char *argv[]){
5     int rank, size;
6     MPI_Init(&argc, &argv);
7     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
8     MPI_Comm_size(MPI_COMM_WORLD, &size);
9     char str[] = "Hello";
10    printf("Process rank- %d\n", rank);
11    if (islower(str[rank])){
12        printf("Converting %c to %c\n", str[rank], toupper(str[rank]));
13    }
14    else{
15        printf("Converting %c to %c\n", str[rank], tolower(str[rank]));
16    }
17    MPI_Finalize();
18    return 0;
19 }
```

Student@dblab-hp-25: ~/180905134/Lab 1

Student@dblab-hp-25: ~/180905134/Lab 1\$ mpicc -o fourth fourth.c

Student@dblab-hp-25: ~/180905134/Lab 1\$ mpirun -n 5 ./fourth

Process rank- 0
Toggling character at 0 index
Converting H to h

Process rank- 1
Toggling character at 1 index
Converting e to E

Process rank- 2
Toggling character at 2 index
Converting L to l

Process rank- 4
Toggling character at 4 index
Converting o to O

Process rank- 3
Toggling character at 3 index
Converting l to L

Student@dblab-hp-25: ~/180905134/Lab 1\$

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:

```
1 #include "mpi.h"
2 #include<stdio.h>
3 #include<math.h>
4 int main(int argc,char *argv[]){
5     int rank,size;
6     MPI_Init(&argc,&argv);
7     MPI_Comm_rank(MPI_COMM_WORLD,&rank);
8     MPI_Comm_size(MPI_COMM_WORLD,&size);
9     int arr[9]={18,11,5,3,2,2,5,6,3};
10    int n=arr[rank];
11    int rev=0;
12    while(n!=0){
13        remainder=n%10;
14        rev=rev*10+remainder;
15        n/=10;
16    }
17    printf("Original number %d\n",n);
18    printf("Reversed number %d\n",rev);
19    MPI_Finalize();
20    return 0;
21 }
```

Student@dblab-hp-25: ~/180905134/Lab 1 \$ mpicc -o add1 add1.c

Student@dblab-hp-25: ~/180905134/Lab 1 \$ mpirun -n 9 ./add1

Original number 18
Reversed number 81
Original number 11
Reversed number 11
Original number 5
Reversed number 5
Original number 3
Reversed number 3
Original number 2
Reversed number 2
Original number 2
Reversed number 2
Original number 5
Reversed number 5
Original number 6
Reversed number 6
Original number 3
Reversed number 3
Student@dblab-hp-25: ~/180905134/Lab 1 \$

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:

