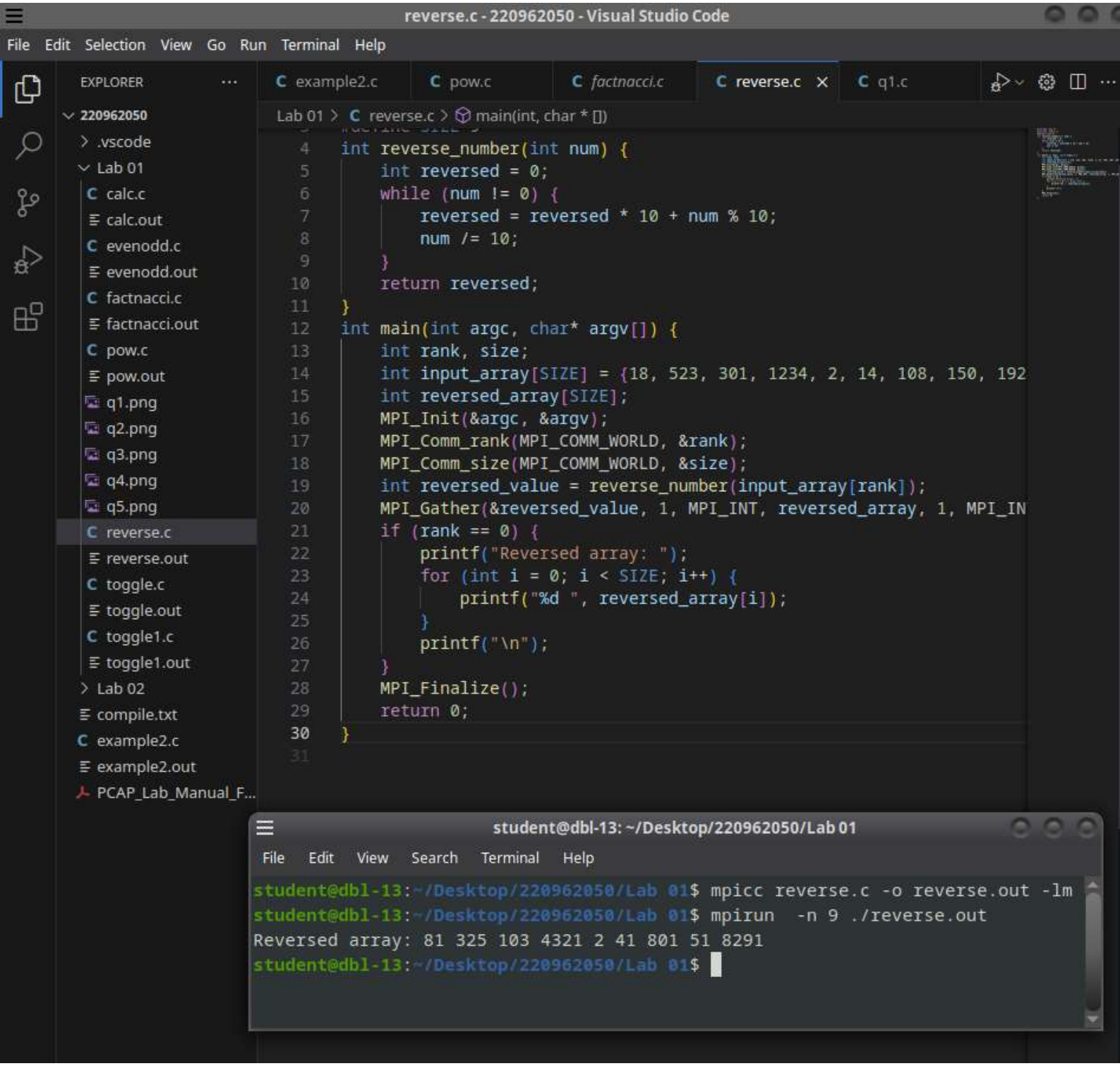
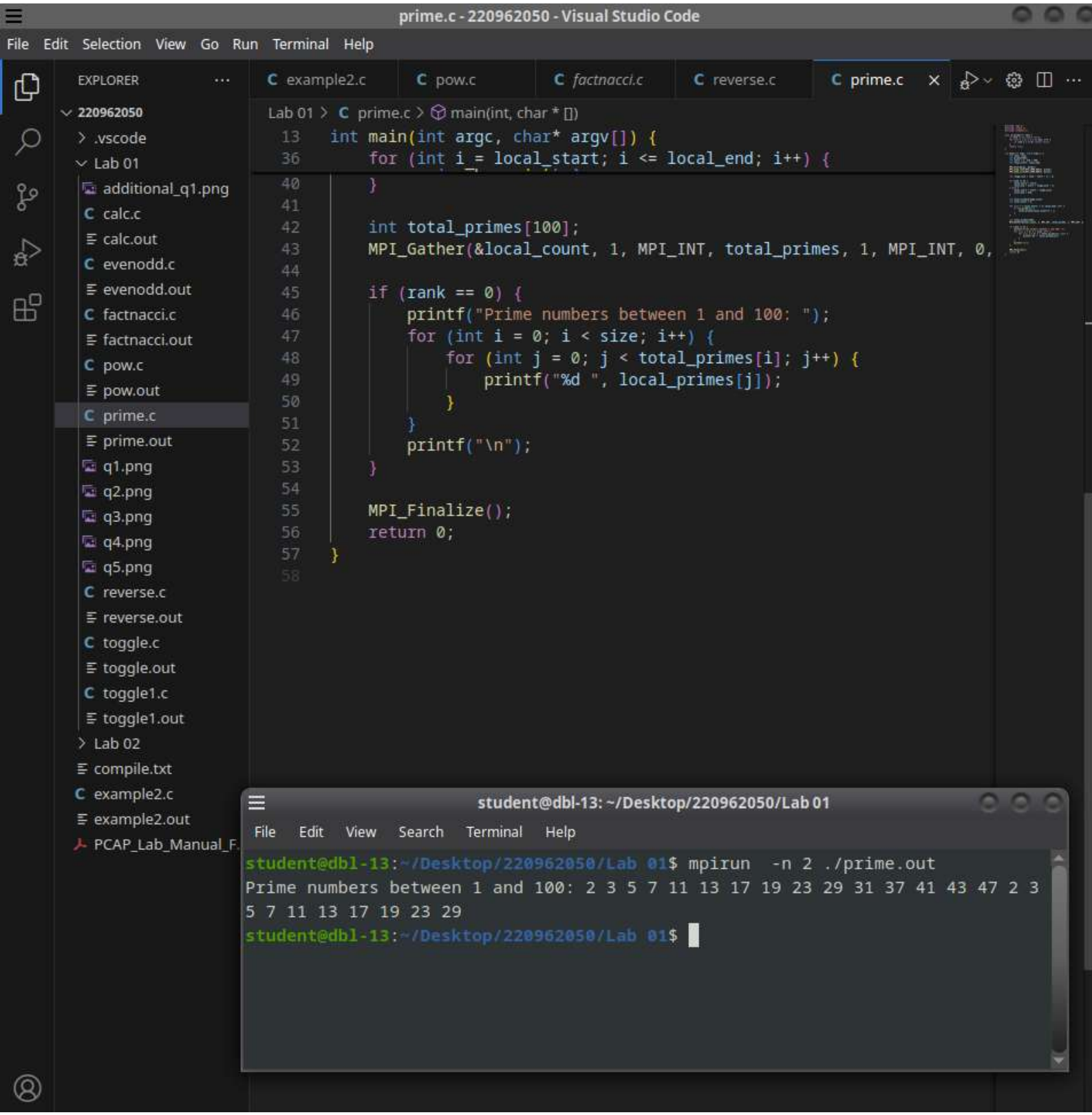
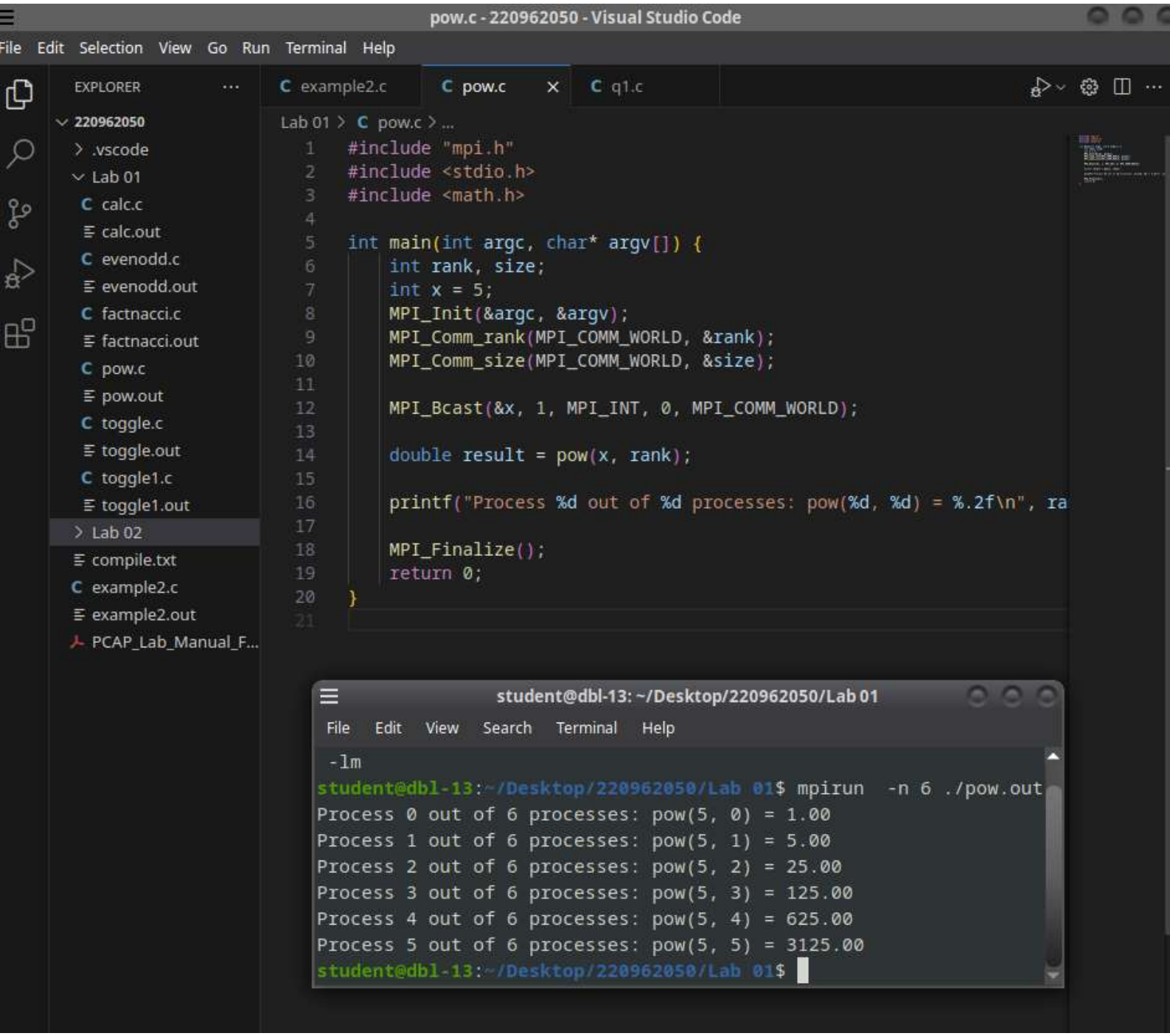
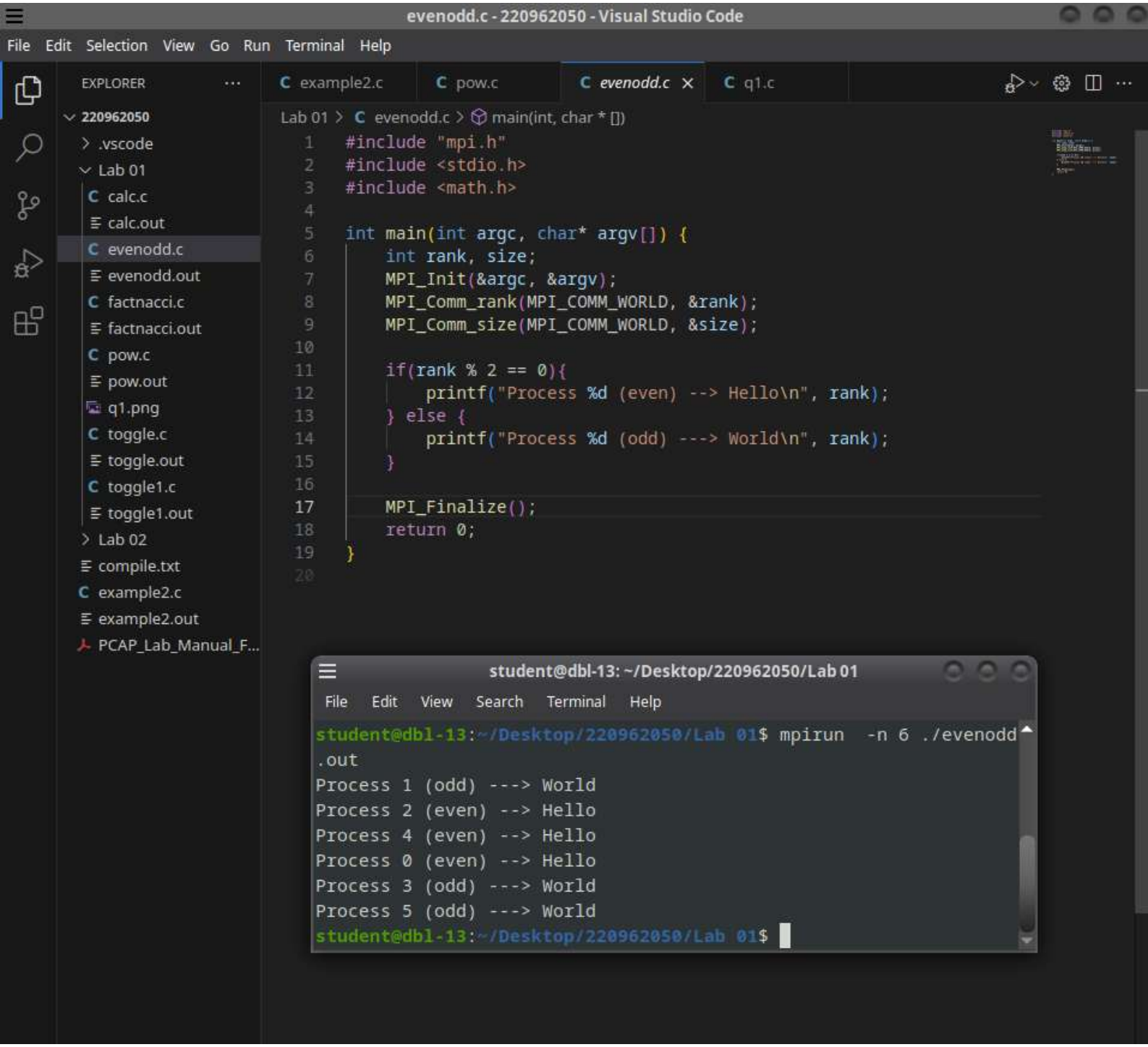

reverse.c - 220962050 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
220962050
.vscode
Lab 01
calc.c
calc.out
evenodd.c
evenodd.out
factnacci.c
factnacci.out
pow.c
pow.out
q1.png
q2.png
q3.png
q4.png
q5.png
reverse.c
reverse.out
toggle.c
toggle.out
toggle1.c
toggle1.out
Lab 02
compile.txt
example2.c
example2.out
PCAP_Lab_Manual_F...
example2.c
pow.c
factnacci.c
reverse.c
q1.c
Lab 01 > reverse.c > main(int, char * [])
int reverse_number(int num) {
int reversed = 0;
while (num != 0) {
reversed = reversed * 10 + num % 10;
num /= 10;
}
return reversed;
}
int main(int argc, char* argv[]) {
int rank, size;
int input_array[SIZE] = {18, 523, 301, 1234, 2, 14, 108, 150, 192
int reversed_array[SIZE];
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
int reversed_value = reverse_number(input_array[rank]);
MPI_Gather(&reversed_value, 1, MPI_INT, reversed_array, 1, MPI_IN
if (rank == 0) {
printf("Reversed array: ");
for (int i = 0; i < SIZE; i++) {
printf("%d ", reversed_array[i]);
}
printf("\n");
}
MPI_Finalize();
return 0;
}
student@dbl-13: ~/Desktop/220962050/Lab 01
File Edit View Search Terminal Help
student@dbl-13:~/Desktop/220962050/Lab 01$ mpicc reverse.c -o reverse.out -lm
student@dbl-13:~/Desktop/220962050/Lab 01$ mpirun -n 9 ./reverse.out
Reversed array: 81 325 103 4321 2 41 801 51 8291
student@dbl-13:~/Desktop/220962050/Lab 01$


prime.c - 220962050 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
220962050
.vscode
Lab 01
additional_q1.png
calc.c
calc.out
evenodd.c
evenodd.out
factnacci.c
factnacci.out
pow.c
pow.out
prime.c
prime.out
q1.png
q2.png
q3.png
q4.png
q5.png
reverse.c
reverse.out
toggle.c
toggle.out
toggle1.c
toggle1.out
Lab 02
compile.txt
example2.c
example2.out
PCAP_Lab_Manual_F...
example2.c
pow.c
factnacci.c
reverse.c
prime.c
Lab 01 > prime.c > main(int, char * [])
13 int main(int argc, char* argv[]) {
36 for (int i = local_start; i <= local_end; i++) {
40 }
41
42 int total_primes[100];
43 MPI_Gather(&local_count, 1, MPI_INT, total_primes, 1, MPI_INT, 0,
44
45 if (rank == 0) {
46 printf("Prime numbers between 1 and 100: ");
47 for (int i = 0; i < size; i++) {
48 for (int j = 0; j < total_primes[i]; j++) {
49 printf("%d ", local_primes[j]);
50 }
51 }
52 printf("\n");
53 }
54
55 MPI_Finalize();
56 return 0;
57 }
58
student@dbl-13: ~/Desktop/220962050/Lab 01
File Edit View Search Terminal Help
student@dbl-13:~/Desktop/220962050/Lab 01$ mpirun -n 2 ./prime.out
Prime numbers between 1 and 100: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 2 3
5 7 11 13 17 19 23 29
student@dbl-13:~/Desktop/220962050/Lab 01$


pow.c - 220962050 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
220962050
.vscode
Lab 01
calc.c
calc.out
evenodd.c
evenodd.out
factnacci.c
factnacci.out
pow.c
pow.out
toggle.c
toggle.out
toggle1.c
toggle1.out
Lab 02
compile.txt
example2.c
example2.out
PCAP_Lab_Manual_F...
example2.c
pow.c
q1.c
Lab 01 > pow.c > ...
#include "mpi.h"
#include <stdio.h>
#include <math.h>
int main(int argc, char* argv[]) {
int rank, size;
int x = 5;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
double result = pow(x, rank);
printf("Process %d out of %d processes: pow(%d, %d) = %.2f\n", ra
MPI_Finalize();
return 0;
}
student@dbl-13: ~/Desktop/220962050/Lab 01
File Edit View Search Terminal Help
-lm
student@dbl-13:~/Desktop/220962050/Lab 01$ mpirun -n 6 ./pow.out
Process 0 out of 6 processes: pow(5, 0) = 1.00
Process 1 out of 6 processes: pow(5, 1) = 5.00
Process 2 out of 6 processes: pow(5, 2) = 25.00
Process 3 out of 6 processes: pow(5, 3) = 125.00
Process 4 out of 6 processes: pow(5, 4) = 625.00
Process 5 out of 6 processes: pow(5, 5) = 3125.00
student@dbl-13:~/Desktop/220962050/Lab 01$


evenodd.c - 220962050 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
220962050
.vscode
Lab 01
calc.c
calc.out
evenodd.c
evenodd.out
factnacci.c
factnacci.out
pow.c
pow.out
q1.png
toggle.c
toggle.out
toggle1.c
toggle1.out
Lab 02
compile.txt
example2.c
example2.out
PCAP_Lab_Manual_F...
example2.c
pow.c
evenodd.c
q1.c
Lab 01 > evenodd.c > main(int, char * [])
#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 % 2 == 0){
printf("Process %d (even) --> Hello\n", rank);
} else {
printf("Process %d (odd) ---> World\n", rank);
}
MPI_Finalize();
return 0;
}
student@dbl-13: ~/Desktop/220962050/Lab 01
File Edit View Search Terminal Help
student@dbl-13:~/Desktop/220962050/Lab 01$ mpirun -n 6 ./evenodd
.out
Process 1 (odd) ---> World
Process 2 (even) --> Hello
Process 4 (even) --> Hello
Process 0 (even) --> Hello
Process 3 (odd) ---> World
Process 5 (odd) ---> World
student@dbl-13:~/Desktop/220962050/Lab 01$

calc.c - 220962050 - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

220962050
- .vscode
- Lab 01
  - calc.c
  - calc.out
  - evenodd.c
  - evenodd.out
  - factnacci.c
  - factnacci.out
  - pow.c
  - pow.out
  - q1.png
  - q2.png
  - q4.png
  - toggle.c
  - toggle.out
  - toggle1.c
  - toggle1.out
- Lab 02
- compile.txt
- example2.c
- example2.out
- PCAP_Lab_Manual_F...

OUTLINE

TIMELINE

example2.c | pow.c | calc.c | q1.c

Lab 01 > calc.c > main(int, char * [])

```c
#include "mpi.h"
#include <stdio.h>

int main(int argc, char* argv[]) {
    int rank, size;
    int num1, num2, result;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (size < 4) {
        if (rank == 0) {
            printf("TRequirement Atleast 4 Processes.\n");
        }
        MPI_Finalize();
        return 0;
    }

    if (rank == 0) {
        printf("Enter Number 1: ");
        scanf("%d",&num1);
        printf("Enter Number 2: ");
        scanf("%d",&num2);
    }

    MPI_Bcast(&num1, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&num2, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        result = num1 + num2;
        printf("Process %d: Addition result ( %d + %d ) = %d\n", rank
    } else if (rank == 1) {
        result = num1 - num2;
        printf("Process %d: Subtraction result ( %d - %d ) = %d\n", r
    } else if (rank == 2) {
        result = num1 * num2;
        printf("Process %d: Multiplication result ( %d * %d ) = %d\n"
    } else if (rank == 3) {
        if (num2 != 0) {
            result = num1 / num2;
            printf("Process %d: Division result ( %d / %d ) = %d\n",
        } else {
            printf("Process %d: Division by zero error.\n", rank);
        }
    }
    MPI_Finalize();
    return 0;
```

Ln 27, Col 53 Spaces: 4 UTF-8 LF {} C Linux

PCAP_Lab_Manual_Final.pdf

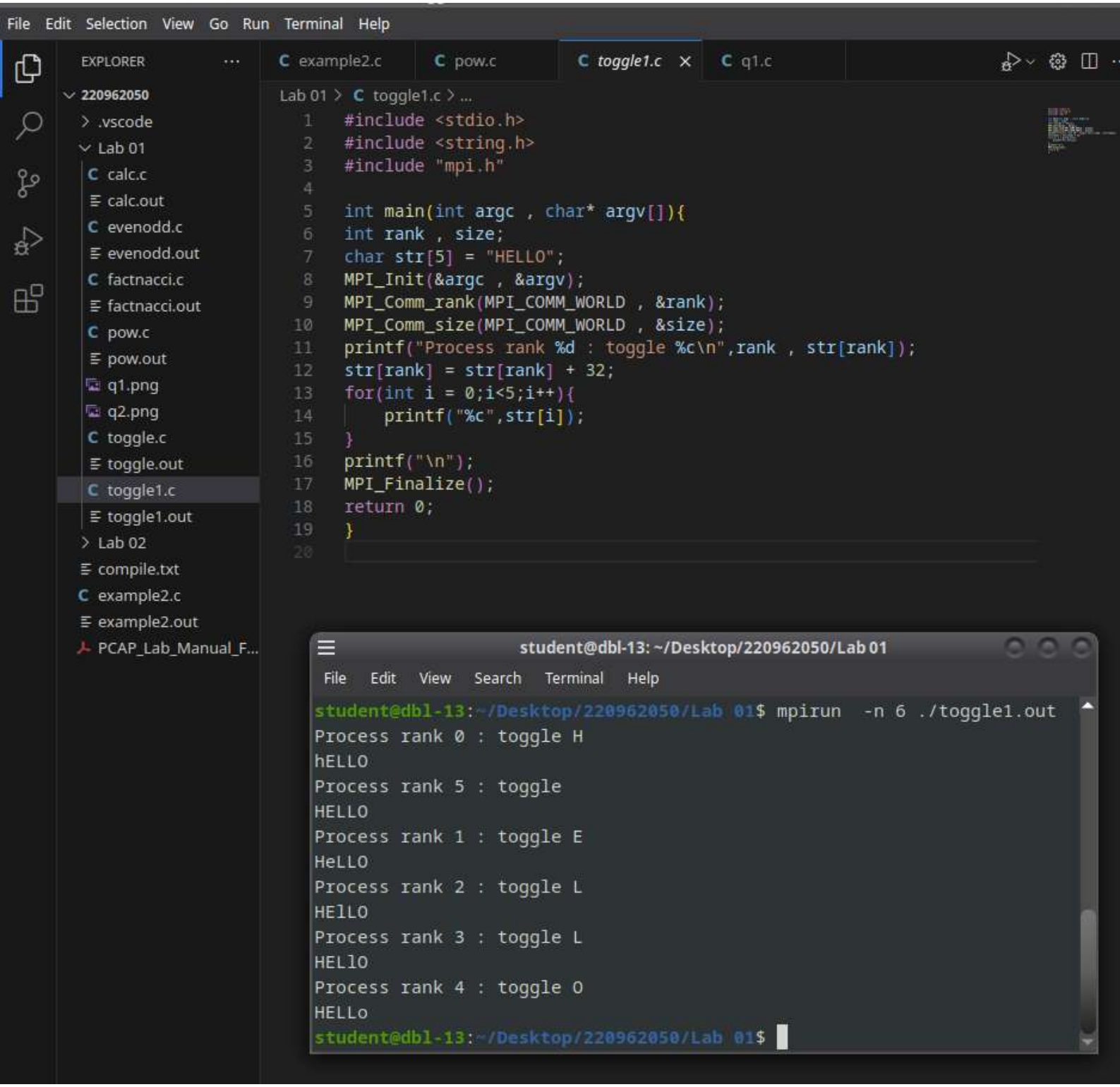
student@dbl-13: ~/Desktop/220962050/Lab 01

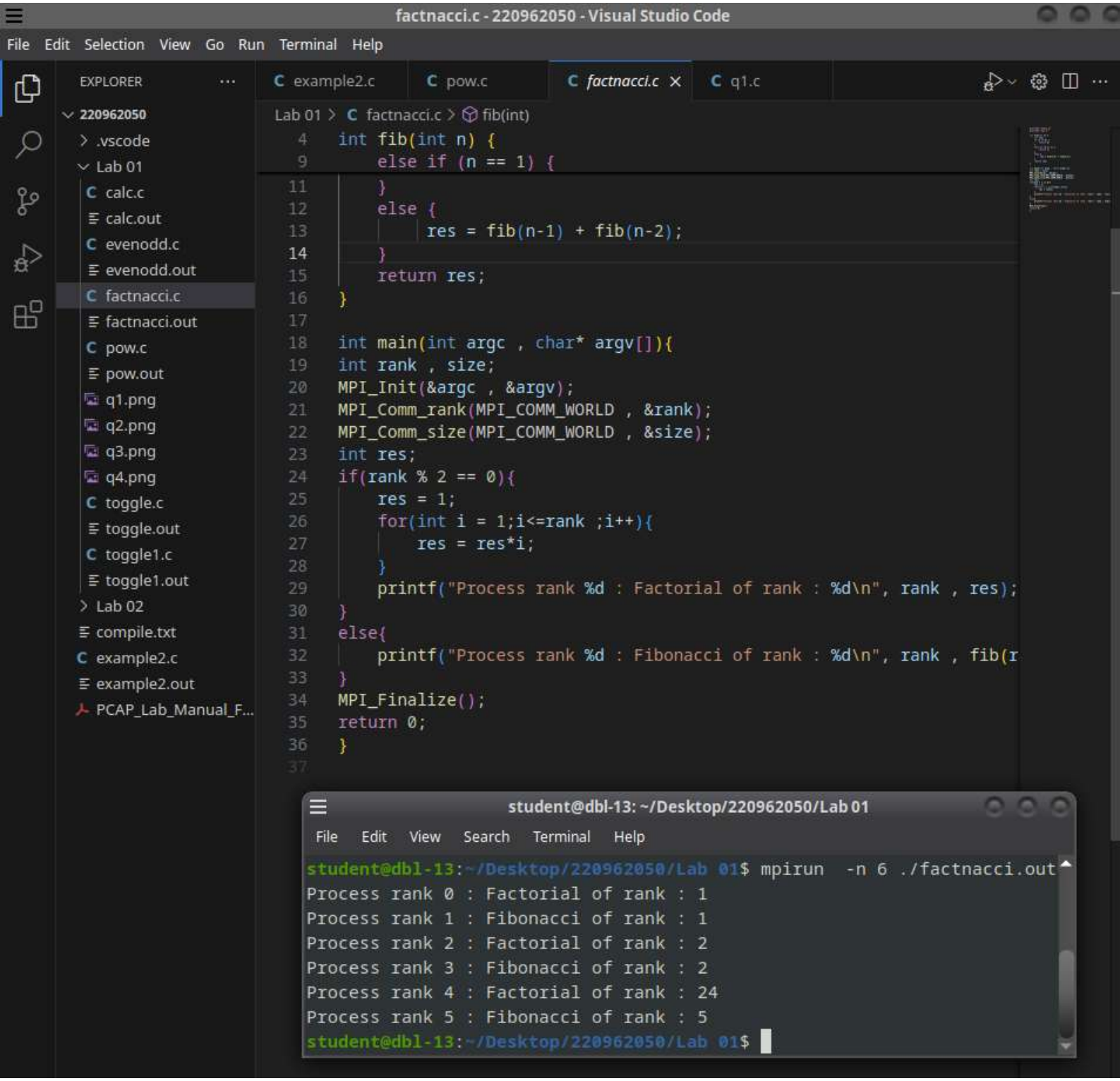
File Edit View Search Terminal Help

```
student@dbl-13:~/Desktop/220962050/Lab 01$ mpirun -n 6 ./calc.out
Enter Number 1: 5
Enter Number 2: 7
Process 0: Addition result ( 5 + 7 ) = 12
Process 1: Subtraction result ( 5 - 7 ) = -2
Process 2: Multiplication result ( 5 * 7 ) = 35
Process 3: Division result ( 5 / 7 ) = 0
student@dbl-13:~/Desktop/220962050/Lab 01$
```

Output array: 81, 325, 103, 4321, 2, 41, 801, 51, 8291

2. Write a MPI program to find the prime numbers between 1 and 100 using two processes.


File Edit Selection View Go Run Terminal Help
EXPLORER
220962050
.vscode
Lab 01
calc.c
calc.out
evenodd.c
evenodd.out
factnacci.c
factnacci.out
pow.c
pow.out
q1.png
q2.png
toggle.c
toggle.out
toggle1.c
toggle1.out
Lab 02
compile.txt
example2.c
example2.out
PCAP_Lab_Manual_F...
example2.c
pow.c
toggle1.c
q1.c
Lab 01 > toggle1.c > ...
#include <stdio.h>
#include <string.h>
#include "mpi.h"
int main(int argc , char* argv[]){
int rank , size;
char str[5] = "HELLO";
MPI_Init(&argc , &argv);
MPI_Comm_rank(MPI_COMM_WORLD , &rank);
MPI_Comm_size(MPI_COMM_WORLD , &size);
printf("Process rank %d : toggle %c\n",rank , str[rank]);
str[rank] = str[rank] + 32;
for(int i = 0;i<5;i++){
printf("%c",str[i]);
}
printf("\n");
MPI_Finalize();
return 0;
}
student@dbl-13: ~/Desktop/220962050/Lab 01
File Edit View Search Terminal Help
student@dbl-13:~/Desktop/220962050/Lab 01$ mpirun -n 6 ./toggle1.out
Process rank 0 : toggle H
hELLO
Process rank 5 : toggle
HELLO
Process rank 1 : toggle E
HeLLO
Process rank 2 : toggle L
HElLO
Process rank 3 : toggle L
HELlO
Process rank 4 : toggle O
HELLo
student@dbl-13:~/Desktop/220962050/Lab 01$


factnacci.c - 220962050 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
220962050
.vscode
Lab 01
calc.c
calc.out
evenodd.c
evenodd.out
factnacci.c
factnacci.out
pow.c
pow.out
q1.png
q2.png
q3.png
q4.png
toggle.c
toggle.out
toggle1.c
toggle1.out
Lab 02
compile.txt
example2.c
example2.out
PCAP_Lab_Manual_F...
example2.c
pow.c
factnacci.c
q1.c
Lab 01 > factnacci.c > fib(int)
4 int fib(int n) {
9 else if (n == 1) {
11 }
12 else {
13 res = fib(n-1) + fib(n-2);
14 }
15 return res;
16 }
17
18 int main(int argc , char* argv[]){
19 int rank , size;
20 MPI_Init(&argc , &argv);
21 MPI_Comm_rank(MPI_COMM_WORLD , &rank);
22 MPI_Comm_size(MPI_COMM_WORLD , &size);
23 int res;
24 if(rank % 2 == 0){
25 res = 1;
26 for(int i = 1;i<=rank ;i++){
27 res = res*i;
28 }
29 printf("Process rank %d : Factorial of rank : %d\n", rank , res);
30 }
31 else{
32 printf("Process rank %d : Fibonacci of rank : %d\n", rank , fib(r
33 }
34 MPI_Finalize();
35 return 0;
36 }
37
student@dbl-13: ~/Desktop/220962050/Lab 01
File Edit View Search Terminal Help
student@dbl-13:~/Desktop/220962050/Lab 01$ mpirun -n 6 ./factnacci.out
Process rank 0 : Factorial of rank : 1
Process rank 1 : Fibonacci of rank : 1
Process rank 2 : Factorial of rank : 2
Process rank 3 : Fibonacci of rank : 2
Process rank 4 : Factorial of rank : 24
Process rank 5 : Fibonacci of rank : 5
student@dbl-13:~/Desktop/220962050/Lab 01$