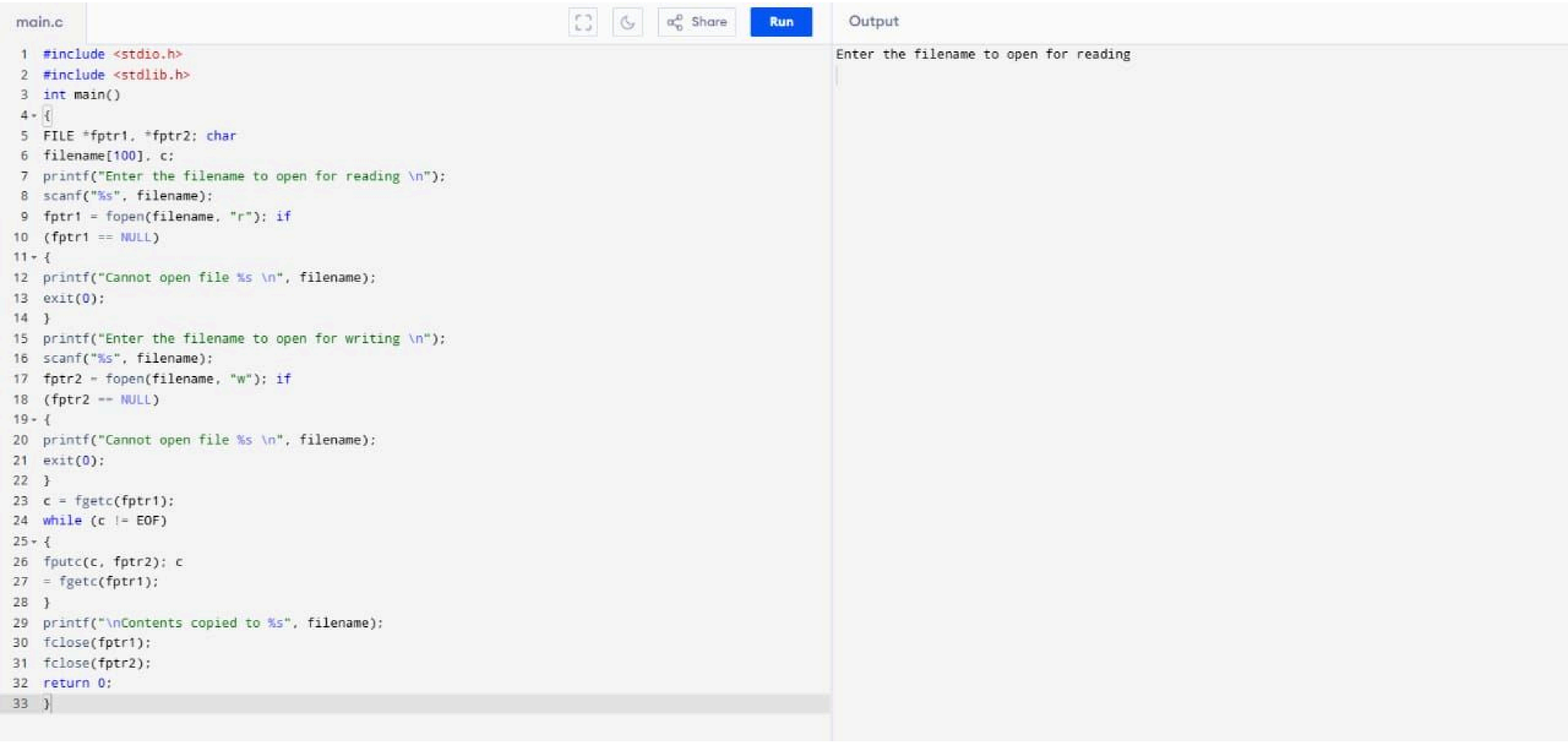
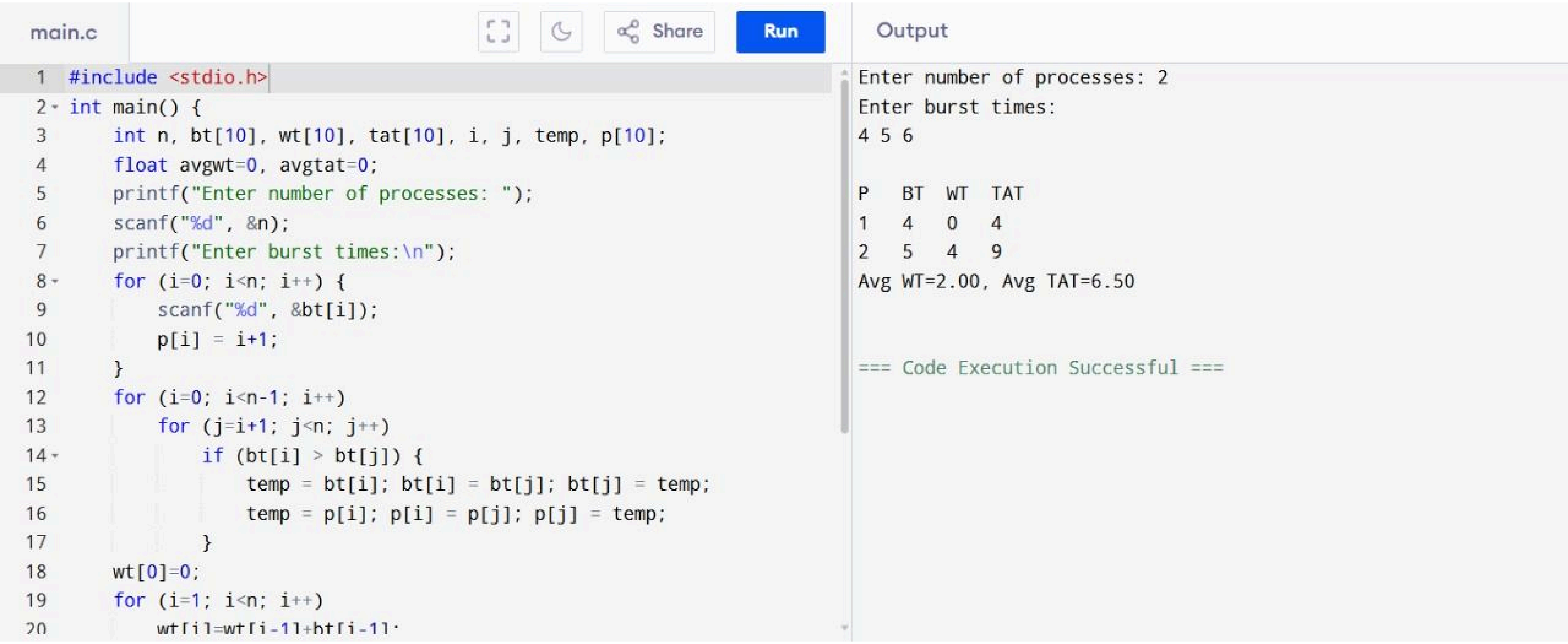


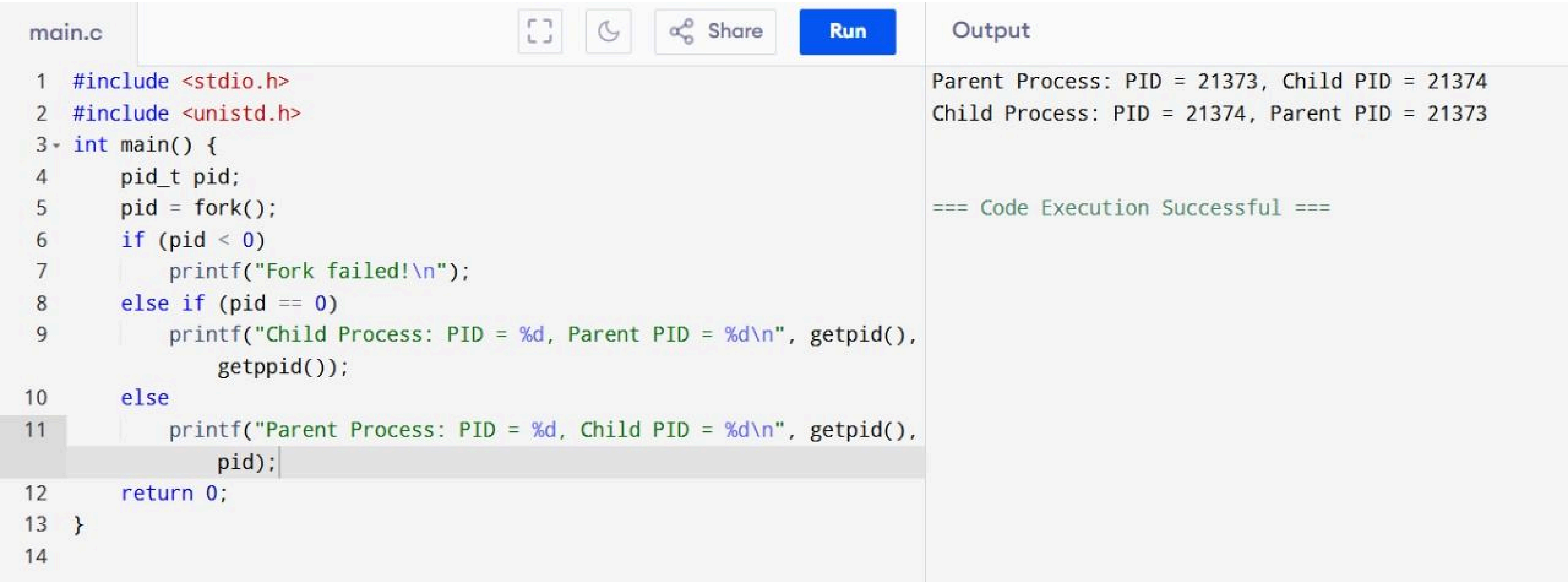
```c
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *fptr1, *fptr2; char
filename[100], c;
printf("Enter the filename to open for reading \n");
scanf("%s", filename);
fptr1 = fopen(filename, "r"); if
(fptr1 == NULL)
{
printf("Cannot open file %s \n", filename);
exit(0);
}
printf("Enter the filename to open for writing \n");
scanf("%s", filename);
fptr2 = fopen(filename, "w"); if
(fptr2 == NULL)
{
printf("Cannot open file %s \n", filename);
exit(0);
}
c = fgetc(fptr1);
while (c != EOF)
{
fputc(c, fptr2); c
= fgetc(fptr1);
}
printf("\nContents copied to %s", filename);
fclose(fptr1);
fclose(fptr2);
return 0;
}
```

```
Enter the filename to open for reading
```



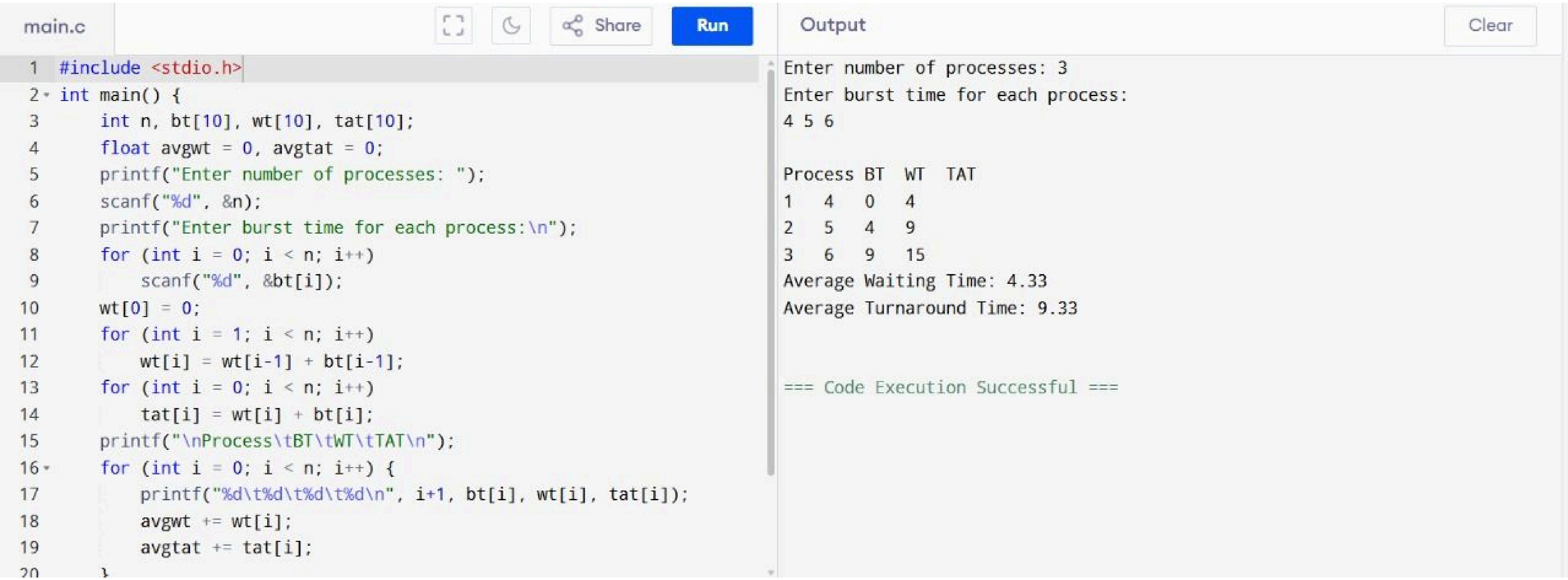
```c
#include <stdio.h>
int main() {
    int n, bt[10], wt[10], tat[10], i, j, temp, p[10];
    float avgwt=0, avgtat=0;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter burst times:\n");
    for (i=0; i<n; i++) {
        scanf("%d", &bt[i]);
        p[i] = i+1;
    }
    for (i=0; i<n-1; i++)
        for (j=i+1; j<n; j++)
            if (bt[i] > bt[j]) {
                temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
                temp = p[i]; p[i] = p[j]; p[j] = temp;
            }
    wt[0]=0;
    for (i=1; i<n; i++)
        wt[i]=wt[i-1]+bt[i-1];
```

```
Enter number of processes: 2
Enter burst times:
4 5 6

P   BT  WT  TAT
1   4   0   4
2   5   4   9
Avg WT=2.00, Avg TAT=6.50


=== Code Execution Successful ===
```



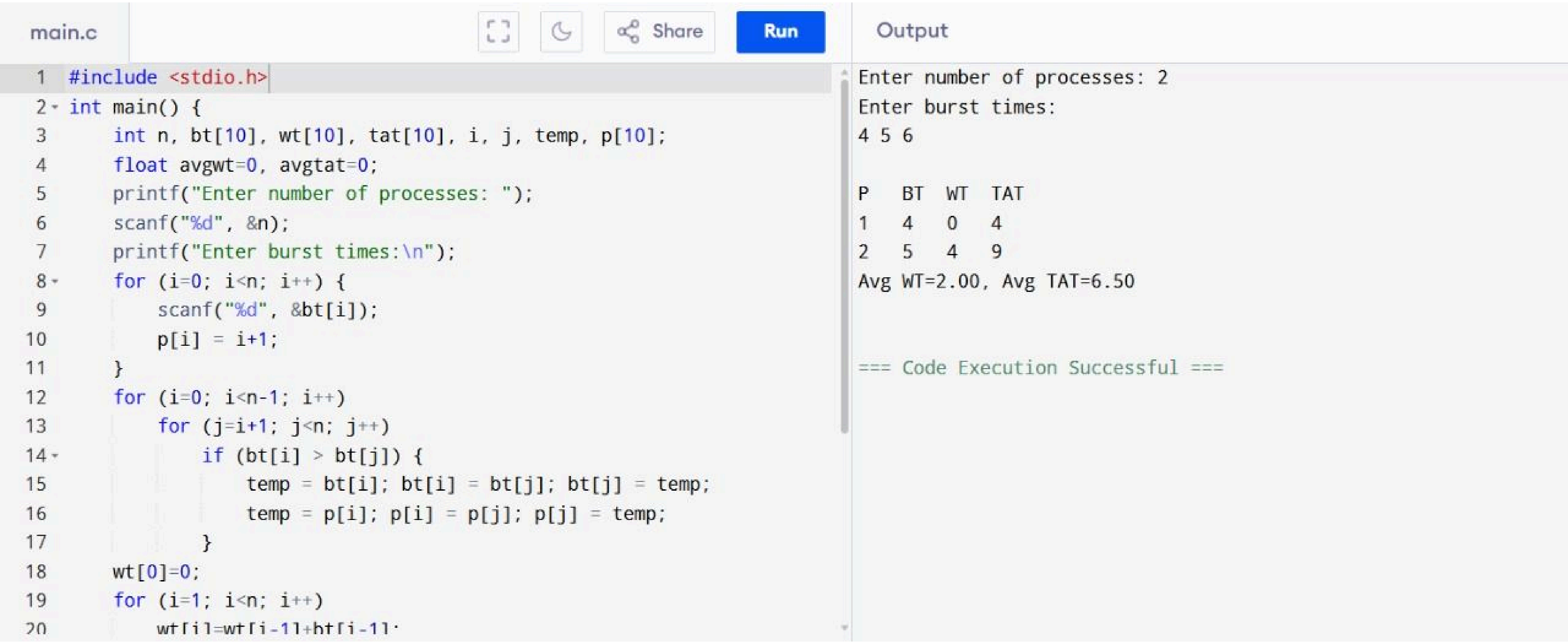
```c
#include <stdio.h>
#include <unistd.h>
int main() {
    pid_t pid;
    pid = fork();
    if (pid < 0)
        printf("Fork failed!\n");
    else if (pid == 0)
        printf("Child Process: PID = %d, Parent PID = %d\n", getpid(),
            getppid());
    else
        printf("Parent Process: PID = %d, Child PID = %d\n", getpid(),
            pid);
    return 0;
}
```

```
Parent Process: PID = 21373, Child PID = 21374
Child Process: PID = 21374, Parent PID = 21373


=== Code Execution Successful ===
```



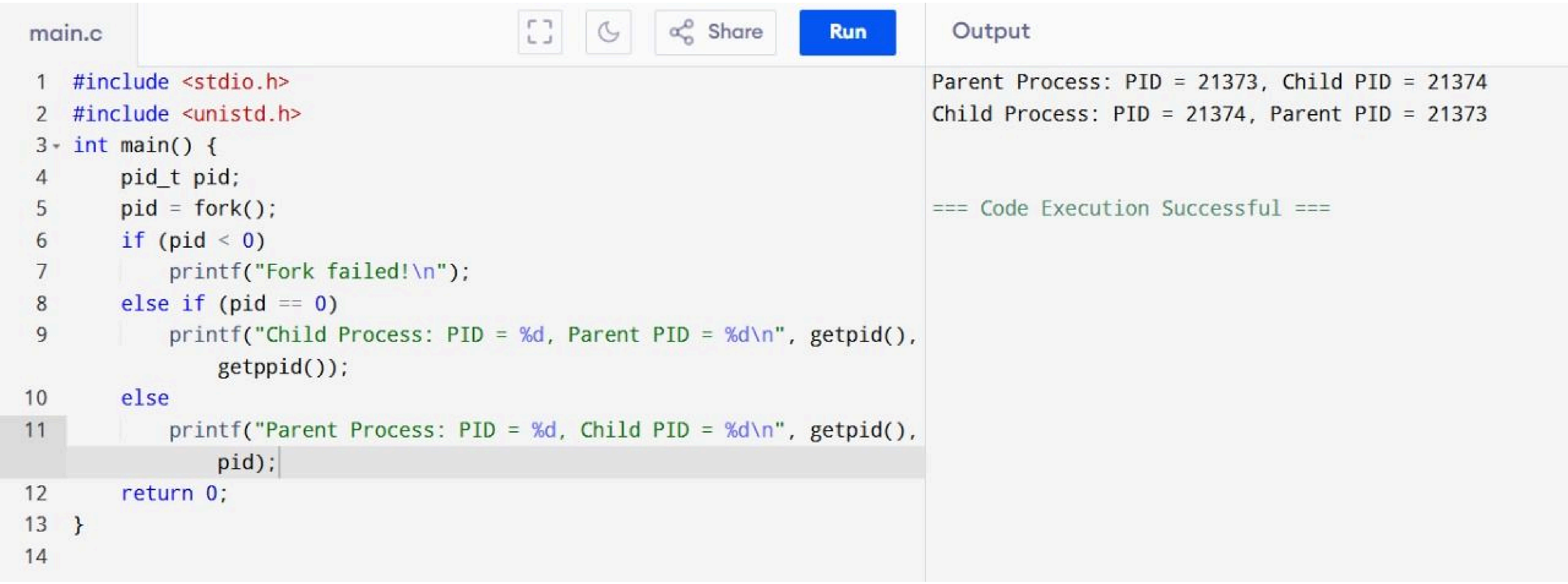
```c
#include <stdio.h>
int main() {
    int n, bt[10], wt[10], tat[10];
    float avgwt = 0, avgtat = 0;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter burst time for each process:\n");
    for (int i = 0; i < n; i++)
        scanf("%d", &bt[i]);
    wt[0] = 0;
    for (int i = 1; i < n; i++)
        wt[i] = wt[i-1] + bt[i-1];
    for (int i = 0; i < n; i++)
        tat[i] = wt[i] + bt[i];
    printf("\nProcess\tBT\tWT\tTAT\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%d\n", i+1, bt[i], wt[i], tat[i]);
        avgwt += wt[i];
        avgtat += tat[i];
    }
```

```
Enter number of processes: 3
Enter burst time for each process:
4 5 6

Process BT  WT  TAT
1    4    0    4
2    5    4    9
3    6    9    15
Average Waiting Time: 4.33
Average Turnaround Time: 9.33


=== Code Execution Successful ===
```



```c
#include <stdio.h>
int main() {
    int n, bt[10], wt[10], tat[10], i, j, temp, p[10];
    float avgwt=0, avgtat=0;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("Enter burst times:\n");
    for (i=0; i<n; i++) {
        scanf("%d", &bt[i]);
        p[i] = i+1;
    }
    for (i=0; i<n-1; i++)
        for (j=i+1; j<n; j++)
            if (bt[i] > bt[j]) {
                temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
                temp = p[i]; p[i] = p[j]; p[j] = temp;
            }
    wt[0]=0;
    for (i=1; i<n; i++)
        wt[i]=wt[i-1]+bt[i-1];
```

```
Enter number of processes: 2
Enter burst times:
4 5 6

P   BT  WT  TAT
1   4   0   4
2   5   4   9
Avg WT=2.00, Avg TAT=6.50


=== Code Execution Successful ===
```



```c
#include <stdio.h>
#include <unistd.h>
int main() {
    pid_t pid;
    pid = fork();
    if (pid < 0)
        printf("Fork failed!\n");
    else if (pid == 0)
        printf("Child Process: PID = %d, Parent PID = %d\n", getpid(),
            getppid());
    else
        printf("Parent Process: PID = %d, Child PID = %d\n", getpid(),
            pid);
    return 0;
}
```

```
Parent Process: PID = 21373, Child PID = 21374
Child Process: PID = 21374, Parent PID = 21373


=== Code Execution Successful ===
```