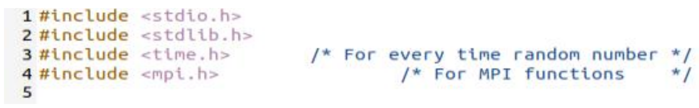
CENG 342 Home Work – 1

Playing a rock paper scissors game with MPI

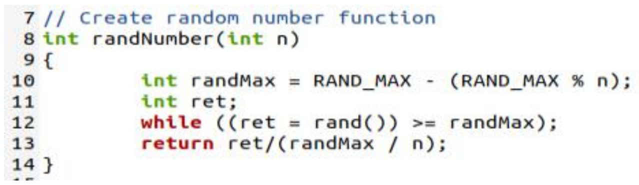
Ömer YILDIRIM 17050161004

1. Import the necessary libraries



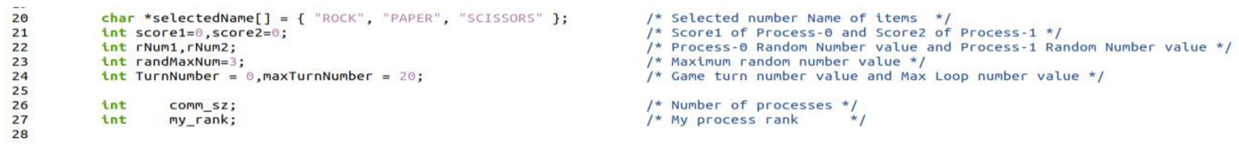
```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>        /* For every time random number */
#include <mpi.h>                /* For MPI functions    */

```

2. Create randım number function



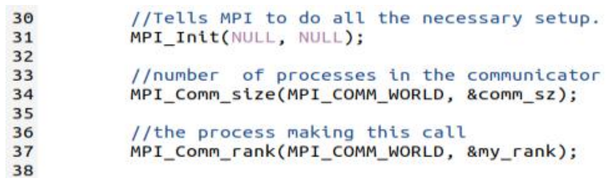
```c
// Create random number function
int randNumber(int n)
{
        int randMax = RAND_MAX - (RAND_MAX % n);
        int ret;
        while ((ret = rand()) >= randMax);
        return ret/(randMax / n);
}
```

3. Add value



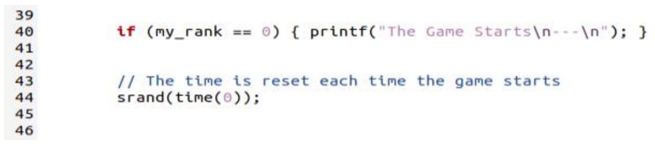
```c
        char *selectedName[] = { "ROCK", "PAPER", "SCISSORS" };         /* Selected number Name of items  */
        int score1=0,score2=0;                                          /* Score1 of Process-0 and Score2 of Process-1 */
        int rNum1,rNum2;                                                /* Process-0 Random Number value and Process-1 Random Number value */
        int randMaxNum=3;                                               /* Maximum random number value */
        int TurnNumber = 0,maxTurnNumber = 20;                          /* Game turn number value and Max Loop number value */

        int     comm_sz;                                                /* Number of processes */
        int     my_rank;                                                /* My process rank      */

```

4. MPI setup



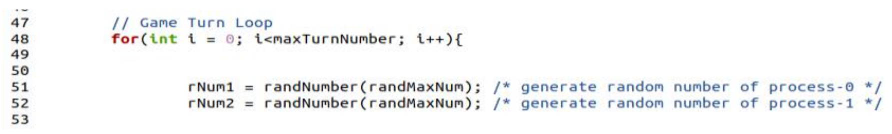
```c
        //Tells MPI to do all the necessary setup.
        MPI_Init(NULL, NULL);

        //number  of processes in the communicator
        MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);

        //the process making this call
        MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

```

5. Start game and Add The time reset to call



```c

        if (my_rank == 0) { printf("The Game Starts\n---\n"); }


        // The time is reset each time the game starts
        srand(time(0));


```

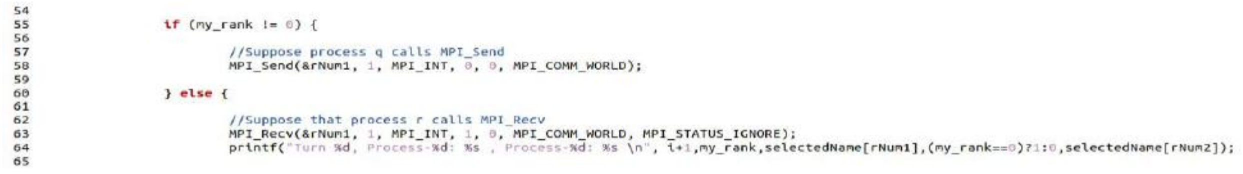
6. Game turn loop and generate random numbers



```c
        // Game Turn Loop
        for(int i = 0; i<maxTurnNumber; i++){


                rNum1 = randNumber(randMaxNum); /* generate random number of process-0 */
                rNum2 = randNumber(randMaxNum); /* generate random number of process-1 */

```

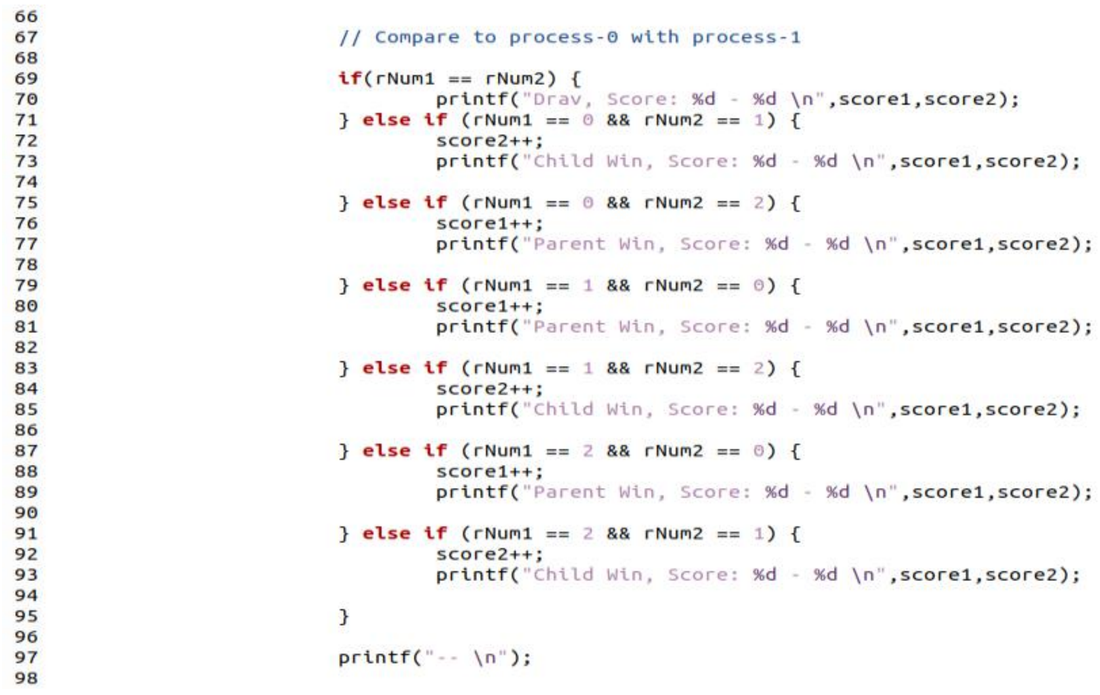
7. MPI send random value with MPI_INT type Without first process

8. MPI receive random value at first process and print two select item



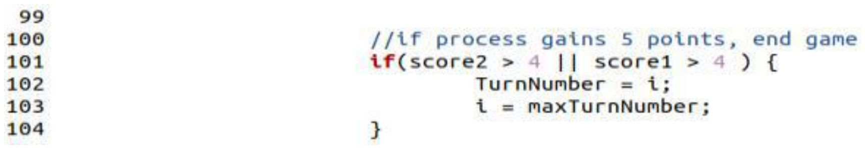
```c

                if (my_rank != 0) {

                        //Suppose process q calls MPI_Send
                        MPI_Send(&rNum1, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);

                } else {

                        //Suppose that process r calls MPI_Recv
                        MPI_Recv(&rNum1, 1, MPI_INT, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
                        printf("Turn %d, Process-%d: %s , Process-%d: %s \n", i+1,my_rank,selectedName[rNum1],(my_rank==0)?1:0,selectedName[rNum2]);

```

9. Compare selecting items



```c

                        // Compare to process-0 with process-1

                        if(rNum1 == rNum2) {
                                printf("Drav, Score: %d - %d \n",score1,score2);
                        } else if (rNum1 == 0 && rNum2 == 1) {
                                score2++;
                                printf("Child Win, Score: %d - %d \n",score1,score2);

                        } else if (rNum1 == 0 && rNum2 == 2) {
                                score1++;
                                printf("Parent Win, Score: %d - %d \n",score1,score2);

                        } else if (rNum1 == 1 && rNum2 == 0) {
                                score1++;
                                printf("Parent Win, Score: %d - %d \n",score1,score2);

                        } else if (rNum1 == 1 && rNum2 == 2) {
                                score2++;
                                printf("Child Win, Score: %d - %d \n",score1,score2);

                        } else if (rNum1 == 2 && rNum2 == 0) {
                                score1++;
                                printf("Parent Win, Score: %d - %d \n",score1,score2);

                        } else if (rNum1 == 2 && rNum2 == 1) {
                                score2++;
                                printf("Child Win, Score: %d - %d \n",score1,score2);

                        }

                        printf("-- \n");

```

10. If process gains 5 points, end game



```c

                        //if process gains 5 points, end game
                        if(score2 > 4 || score1 > 4 ) {
                                TurnNumber = i;
                                i = maxTurnNumber;
                        }
```

11. MPI finish

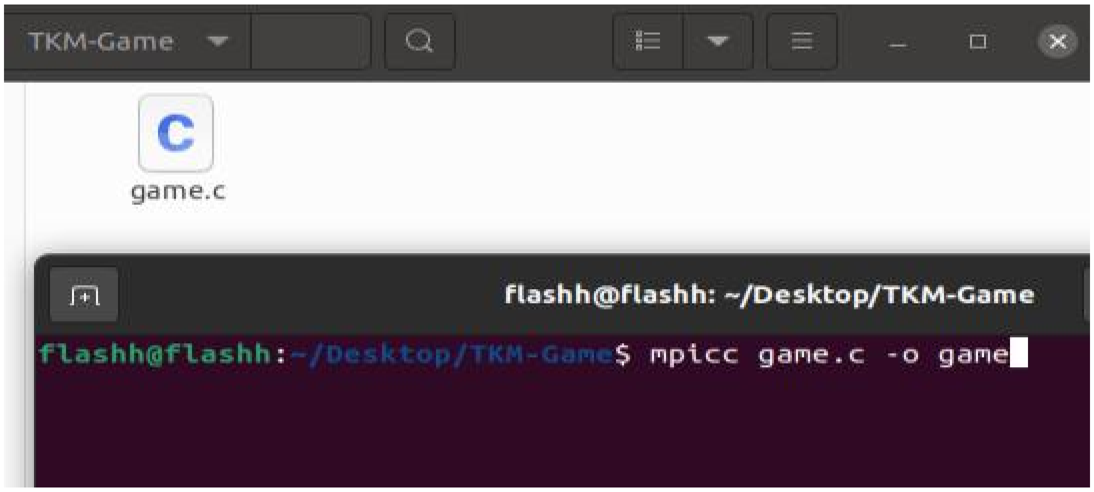
```c
    //Tells MPI we're done, so clean up anything allocated for this program
    MPI_Finalize();

```

11. Determination of the winner and the game ends

```c
    //Game over and determination of the winner
    if (my_rank == 0) {
            if(score1 == score2) {
                    printf("Process-0 and Process-1 have drawn the game with score: %d - %d in %d Turns. \n",score1,score2,TurnNumber);
            } else if(score1 > score2) {
                    printf("Process-0 has won the game with score: %d - %d in %d Turns. \n",score1,score2,TurnNumber);
            } else {
                    printf("Process-1 has won the game with score: %d - %d in %d Turns. \n",score1,score2,TurnNumber);
            }

            printf("\nThe game ends\n\n");
    }

```

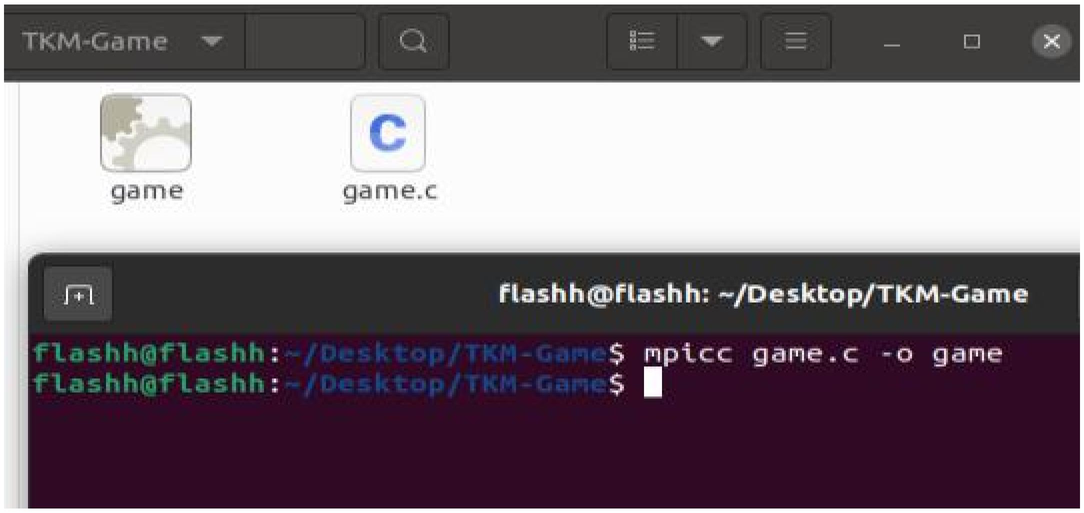
12. Compile Script

>> **mpicc** (*wrapper script to compile*) **game.c** (*source file*) **-o** **game** (*create this executable file name*)



```
flashh@flashh:~/Desktop/TKM-Game$ mpicc game.c -o game
```

13. Created executable file

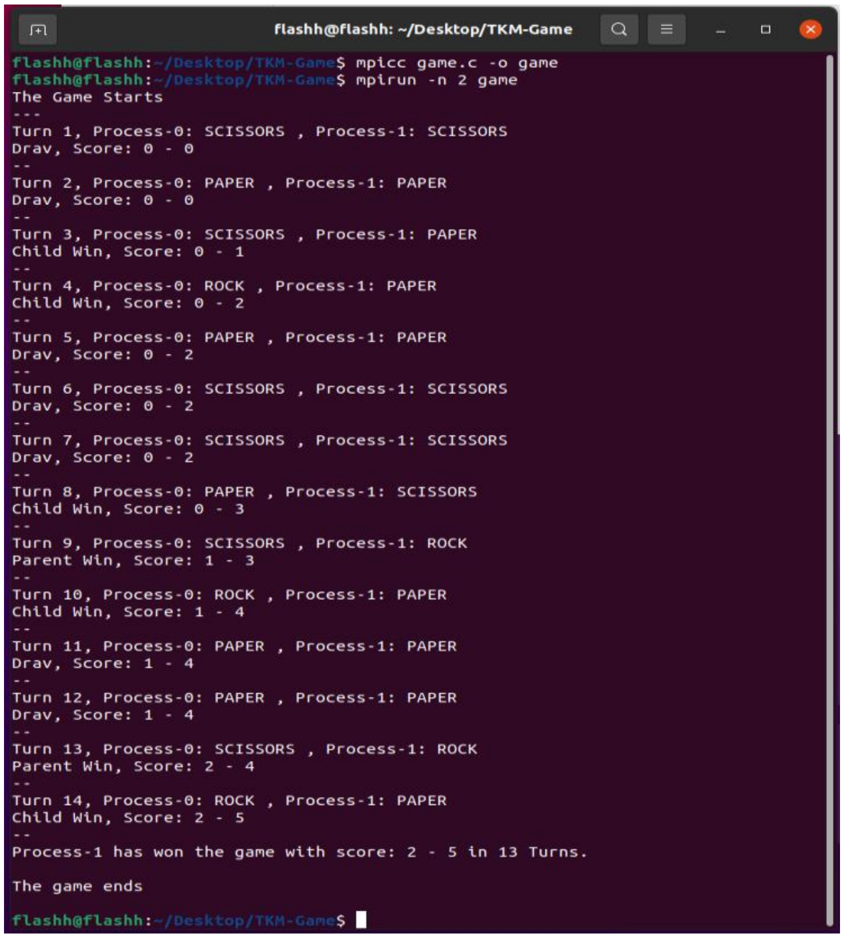


```
flashh@flashh:~/Desktop/TKM-Game$ mpicc game.c -o game
flashh@flashh:~/Desktop/TKM-Game$ 
```

14. Execution

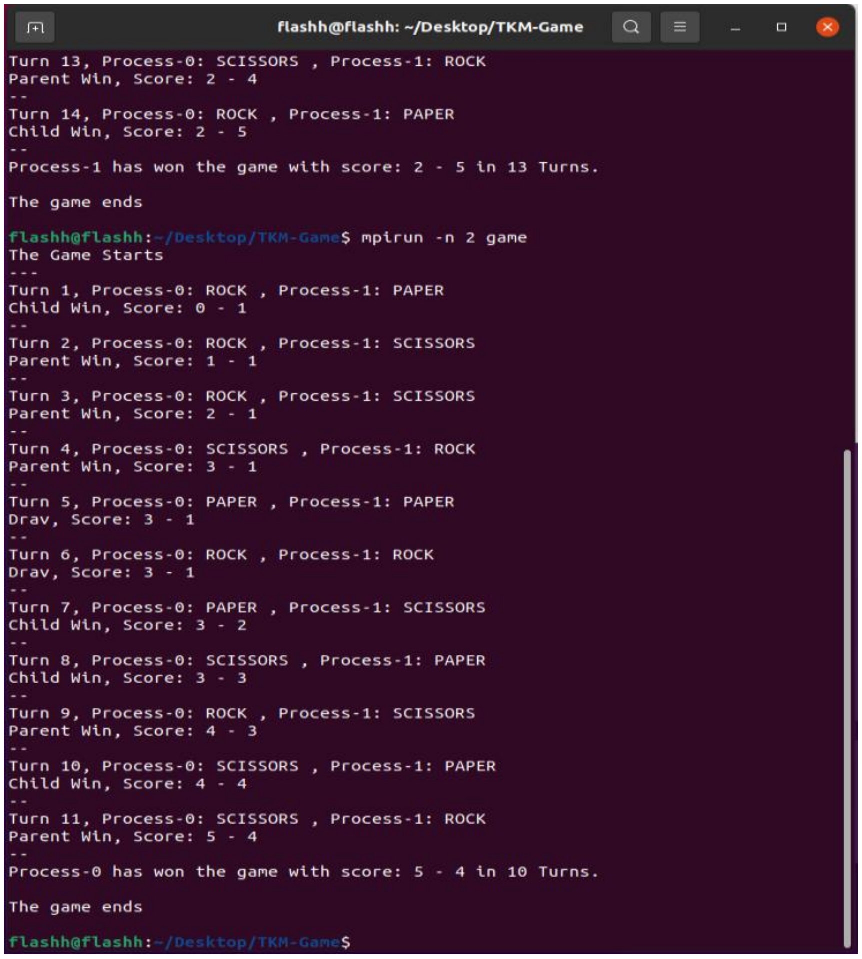
>> mpirun -n <number of processes> <executable>

>> mpirun -n 2 game



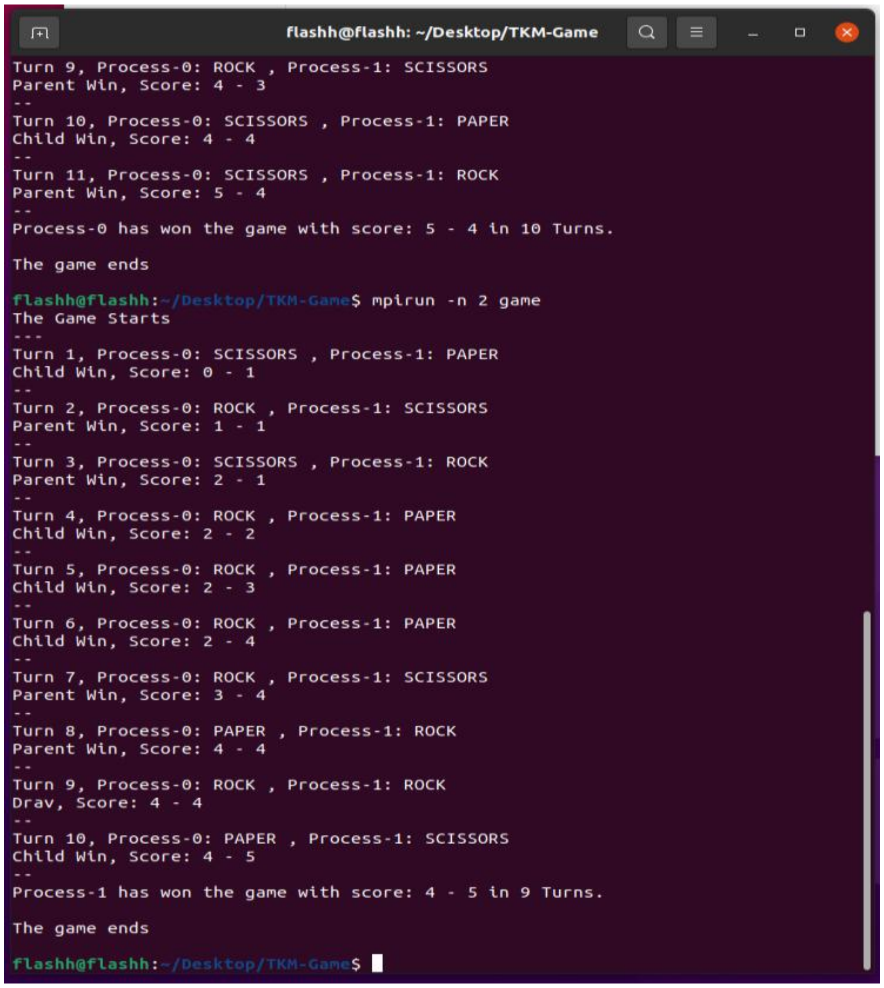
```
flashh@flashh:~/Desktop/TKM-Game$ mpicc game.c -o game
flashh@flashh:~/Desktop/TKM-Game$ mpirun -n 2 game
The Game Starts
---
Turn 1, Process-0: SCISSORS , Process-1: SCISSORS
Drav, Score: 0 - 0
--
Turn 2, Process-0: PAPER , Process-1: PAPER
Drav, Score: 0 - 0
--
Turn 3, Process-0: SCISSORS , Process-1: PAPER
Child Win, Score: 0 - 1
--
Turn 4, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 0 - 2
--
Turn 5, Process-0: PAPER , Process-1: PAPER
Drav, Score: 0 - 2
--
Turn 6, Process-0: SCISSORS , Process-1: SCISSORS
Drav, Score: 0 - 2
--
Turn 7, Process-0: SCISSORS , Process-1: SCISSORS
Drav, Score: 0 - 2
--
Turn 8, Process-0: PAPER , Process-1: SCISSORS
Child Win, Score: 0 - 3
--
Turn 9, Process-0: SCISSORS , Process-1: ROCK
Parent Win, Score: 1 - 3
--
Turn 10, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 1 - 4
--
Turn 11, Process-0: PAPER , Process-1: PAPER
Drav, Score: 1 - 4
--
Turn 12, Process-0: PAPER , Process-1: PAPER
Drav, Score: 1 - 4
--
Turn 13, Process-0: SCISSORS , Process-1: ROCK
Parent Win, Score: 2 - 4
--
Turn 14, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 2 - 5
--
Process-1 has won the game with score: 2 - 5 in 13 Turns.

The game ends

flashh@flashh:~/Desktop/TKM-Game$ 
```

15. Second execution



```
flashh@flashh: ~/Desktop/TKM-Game
Turn 13, Process-0: SCISSORS , Process-1: ROCK
Parent Win, Score: 2 - 4
--
Turn 14, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 2 - 5
--
Process-1 has won the game with score: 2 - 5 in 13 Turns.

The game ends

flashh@flashh:~/Desktop/TKM-Game$ mpirun -n 2 game
The Game Starts
---
Turn 1, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 0 - 1
--
Turn 2, Process-0: ROCK , Process-1: SCISSORS
Parent Win, Score: 1 - 1
--
Turn 3, Process-0: ROCK , Process-1: SCISSORS
Parent Win, Score: 2 - 1
--
Turn 4, Process-0: SCISSORS , Process-1: ROCK
Parent Win, Score: 3 - 1
--
Turn 5, Process-0: PAPER , Process-1: PAPER
Drav, Score: 3 - 1
--
Turn 6, Process-0: ROCK , Process-1: ROCK
Drav, Score: 3 - 1
--
Turn 7, Process-0: PAPER , Process-1: SCISSORS
Child Win, Score: 3 - 2
--
Turn 8, Process-0: SCISSORS , Process-1: PAPER
Child Win, Score: 3 - 3
--
Turn 9, Process-0: ROCK , Process-1: SCISSORS
Parent Win, Score: 4 - 3
--
Turn 10, Process-0: SCISSORS , Process-1: PAPER
Child Win, Score: 4 - 4
--
Turn 11, Process-0: SCISSORS , Process-1: ROCK
Parent Win, Score: 5 - 4
--
Process-0 has won the game with score: 5 - 4 in 10 Turns.

The game ends

flashh@flashh:~/Desktop/TKM-Game$
```

16. Third execution



flashh@flashh: ~/Desktop/TKM-Game

```
Turn 9, Process-0: ROCK , Process-1: SCISSORS
Parent Win, Score: 4 - 3
--
Turn 10, Process-0: SCISSORS , Process-1: PAPER
Child Win, Score: 4 - 4
--
Turn 11, Process-0: SCISSORS , Process-1: ROCK
Parent Win, Score: 5 - 4
--
Process-0 has won the game with score: 5 - 4 in 10 Turns.

The game ends

flashh@flashh:~/Desktop/TKM-Game$ mpirun -n 2 game
The Game Starts
---
Turn 1, Process-0: SCISSORS , Process-1: PAPER
Child Win, Score: 0 - 1
--
Turn 2, Process-0: ROCK , Process-1: SCISSORS
Parent Win, Score: 1 - 1
--
Turn 3, Process-0: SCISSORS , Process-1: ROCK
Parent Win, Score: 2 - 1
--
Turn 4, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 2 - 2
--
Turn 5, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 2 - 3
--
Turn 6, Process-0: ROCK , Process-1: PAPER
Child Win, Score: 2 - 4
--
Turn 7, Process-0: ROCK , Process-1: SCISSORS
Parent Win, Score: 3 - 4
--
Turn 8, Process-0: PAPER , Process-1: ROCK
Parent Win, Score: 4 - 4
--
Turn 9, Process-0: ROCK , Process-1: ROCK
Draw, Score: 4 - 4
--
Turn 10, Process-0: PAPER , Process-1: SCISSORS
Child Win, Score: 4 - 5
--
Process-1 has won the game with score: 4 - 5 in 9 Turns.

The game ends

flashh@flashh:~/Desktop/TKM-Game$
```