Assignment 4. MPI.

Convert the code Assignment4.c to match your individual version of the assignment. Your task number is your number in the reporting table https://docs.google.com/spreadsheets/d/1m2KqmSc5pMMQ9JezZGybnuKG75FywUcs8gbPuMvVUxc. If your number is more than the options for the task, then calculate your number in modulus 28 (for example, 5 (mod 3) = 2).

Note. If the number of processes in the task is not defined, then we can assume that this number does not exceed 8. The main process is understood as a process of rank 0 for the communicator MPI_COMM_WORLD. For all processes of nonzero rank in the tasks, the common name of the subordinate processes is used. If the task does not define the maximum size of a set of numbers, then it should be considered equal to 10.

#1

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your task (1).

Arrange to receive messages not from any processes, but in accordance with the sender's process number.

#2

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your task (2).

Output to the console the information (the rank of the sender process) received by the root process from the other processes.

#3

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your version (3).

Send the int value to the variable equal to the rank value of the sender process, and output the information received by the root process from the other processes to the console.

#4

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your version (4).

Send the double value of the variable equal to the rank value of the sender process and output to the console the information received by the root process from the other processes.

#5

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your version (5).

Send the float value of the variable equal to the rank value of the sender process and output to the console the information received by the root process from the other processes.

#6

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your version (6).

Send the char value of the variable equal to the rank value of the sender process and output to the console the information received by the root process from the other processes.

#7

Each process is given an integer N> 0 and a set of N random numbers. In processes of even rank, output the sum of numbers from a given set; in processes of odd rank, output the arithmetic mean of numbers from a given set.

#8

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your version (8).

Send the value of the variable equal to the value of the rank of the sender process, and output to the console the number received by the root process from the other process, increased by a number equal to the number of your root process.

#9

Receiving messages and outputting information to the console should be carried out by a process with a number that matches the number of your version (8).

Send the value of the variable equal to the value of the rank of the sender process, and output to the console the number received by the root process from the other process, divided by a number equal to the number of your root process.

#10

Process 1 sends a message to process 2, process 2 takes a message from process 1, prints it to the console and sends its message to process 3, and so on.

#11

Process 1 sends a message to process 2, process 2 receives a message from process 1 and sends a message to process 1, process 1 receives a message from process 2 and prints it to the console.

#12

Process 1 sends a message to process 2, process 2 sends a message to process 3, etc., the last process in the group of n processes receives a message from process n-1 and sends a message to the root process, which the latter prints to the console.

#13

Process 1 sends a message to process 2, process 2 prints the message to the console. Process 2 sends a message to process 3, process 3 prints a message to the console, and so on.

#14

The root process sends a message to all processes, all processes output the received information to the console.

#15

The root process sends a message to all processes as a random int value (from 0 to the total number of processes), all processes output the received information to the console if the received value is greater than the rank of the recipient process.

#16

The root process sends a message to all processes as a random int value (from 0 to the total number of processes), all processes output the received information to the console if the received value is less than the rank of the recipient process.

#17

All processes send messages to all processes in the form of a random int value (from 0 to the total number of processes), all processes output the received information to the console if the received value is greater than the rank of the recipient process.

#18

Each process sends its rank number to the root process, the root process squares this number and prints the rank and the result to the console.

#19

The root process takes a sequence of N integers from 5 child processes, calculates the difference between the product of odd numbers and the largest, and prints the result to the console.

#20

The root process accepts messages from child processes and determines whether the sequence is strictly ascending.

#21

The root process accepts messages from child processes and determines whether the sequence is strictly descending.

#22

The root process accepts messages from child processes and determines if the sequence contains at least two adjacent even numbers.

#23

The root process accepts messages from child processes and determines if the sequence contains at least two adjacent odd numbers.

#24

The root process accepts messages from child processes, calculates the product of even numbers, prints the result to the console.

#25

The root process accepts messages from child processes, calculates the product of odd numbers, prints the result to the console.

#26

In the main process, a set of real numbers is given; the number of numbers is equal to the number of slave processes. Using the MPI_Send function, send one number to each of the slave processes (the first number to process 1, the second to process 2, etc.) and output the received numbers in the slave processes.

#27

Each subprocess is given four integers. Send these numbers to the main process using one call to the MPI_Send function for each sending process, and display them in the main process. The resulting numbers should be displayed in ascending order of the ranks of the processes that sent them.

#28

Each slave process is given an integer, and only for one process this number is nonzero. Send a nonzero number to the main process and print the received number and the rank of the process that sent this number in the main process. To receive a message in the main process, use the MPI_Recv function with the MPI_ANY_SOURCE parameter.