```
#include <stdio.h>
#include <mpi.h>
int main(int argc,char **argv)
{
int rank, size, rc;
MPI_Comm comm;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
printf("size = %d process = %d\n",size,rank);
rc=MPI_Finalize();
return 0;
}
```

Задача 1
Процесс 0-й передает значение **x** процессу size-1

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Send_Recv
int main(int argc,char **argv)
{
int rank, size, rc, tag=6;  float x;
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
if(rank==0)
{
x=(float)rand()/RAND_MAX;
rc=MPI_Send(&x, 1, MPI_FLOAT, size-1, tag, comm);
cout <<"process 0 Send x = "<< x << endl;
}
else if(rank==size-1)
{
rc=MPI_Recv(&x, 1, MPI_FLOAT, 0, tag, comm, &status);
cout<<"process = "<< rank <<"  Recv x="<< x <<endl;
}
else
{   cout <<"  process = "<< rank <<" no work !";  }
rc=MPI_Finalize();
return 0;
}
```

Задача 2
Процесс 0-й передает **x** процессу 1, 2, …, size-1

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Send_Recv
int main(int argc,char **argv)
{
int rank, size, rc, tag=6;   float x;
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
if(rank==0)
{
x=(float)rand()/RAND_MAX;
for(int k=1; k<size; k++)
rc=MPI_Send(&x, 1, MPI_FLOAT, k, tag, comm);
cout <<"process 0 Send x = "<< x << endl;;
}
else
{
rc=MPI_Recv(&x, 1, MPI_FLOAT, 0, tag, comm, &status);
cout<<"process = "<< rank <<"  Recv x="<< x << endl;
}
rc=MPI_Finalize();
return 0;
}
```

Задача 3
Процесс 0-й передает компоненты массива **x**[1] -> 1, **x**[2] -> 2, …, **x**[size-1] -> size-1

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Send_Recv
main(int argc,char **argv)
{
int rank, size, rc,t ag=6, n=10;
float x[n],y;
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
if(rank==0)
{
for(int k=0; k<n; k++) x[k]=k*10;
for(int k=1; k<size; k++)
  {
rc=MPI_Send(&x[k], 1, MPI_FLOAT, k, tag, comm);
cout <<"process 0 Send x = "<< x[k] << endl;
  }
}
else
{
rc=MPI_Recv(&y, 1, MPI_FLOAT, 0, tag, comm, &status);
cout<<"process = "<< rank <<"  Recv y="<< y<< endl;
}
rc=MPI_Finalize();
return 0;
}
```

Задача 4
Процесс 0-й принимает **x** от процесса 1, от процесса 2, …, от процесса size-1

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Send_Recv
int main(int argc,char **argv)
{
int rank, size, rc, tag=6;
float x;
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc,&argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
if( rank == 0 )
{
for(int k=1; k<size; k++)
  {
rc=MPI_Recv(&x, 1, MPI_FLOAT, k, tag, comm, &status);
cout <<"process 0 Recv x = "<< x <<" from rank = "<< k <<endl;
   }
}
else
{
x=(float)rand()/RAND_MAX*rank;
rc=MPI_Send(&x, 1, MPI_FLOAT, 0, tag, comm);
cout <<"process = "<< rank <<"  Send x="<< x << endl;
}
rc=MPI_Finalize();
return 0;
}
```

Задача 5
Процесс 0-й принимает **x** от процесса 1, от процесса 2, …, от процесса size-1 в произвольном порядке

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Send_Recv
int main(int argc,char **argv)
{
int rank, size, rc, tag=8;
float x;
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
if( rank == 0 )
{
for(int k=1; k<size; k++)
  {
rc=MPI_Recv(&x,1,MPI_FLOAT,MPI_ANY_SOURCE,MPI_ANY_TAG,comm,&status);
cout <<"process 0 Recv x = "<< x <<" from rank = "<< k << endl;
  }
}
else
{
x=(float)rand()/RAND_MAX*rank;
rc=MPI_Send(&x, 1, MPI_FLOAT, 0, tag, comm);
cout <<"process = "<< rank <<"  Send x="<< x << endl;
}
rc=MPI_Finalize();
return 0;
}
```

Задача 6
Процесс 0 передает вещественный массив **x** процессу size-1 и принимает от него целочисленный массив **y**; процесс size-1 принимает от процесса 0 массив **x** и передает массив **y**.
(функции MPI_Send и MPI_Recv)

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Send_Recv
int main(int argc,char **argv)
{
int rank, size, rc, tag1=6, n1=5, n2=3, tag2=0;
float x[n1];  int y[n2];
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
if(rank==0)
{
for(int j=0; j<=10*(rank+1); j++)
  {  for(int k=0; k<n1; k++)
      { x[k]=(float)rand()/RAND_MAX;  }   }
rc=MPI_Send(&x, n1, MPI_FLOAT, size-1, tag1, comm);
cout <<"process 0 Send x = ";
for(int k=0; k<n1; k++) cout << x[k] <<"  ";
cout << endl;
rc=MPI_Recv(&y, n2, MPI_INT,s ize-1, tag2, comm, &status);
cout <<"process 0 Recv y = ";
for(int k=0; k<n2; k++) cout << y[k] <<"  ";
cout << endl;
}
else if(rank==size-1)
{
rc=MPI_Recv(&x, n1, MPI_FLOAT, 0, tag1, comm, &status);
cout<<"process "<< rank <<"  Recv x = ";
for(int k=0; k<n1; k++) cout << x[k] <<"  ";
cout << endl;
for(int k=0; k<n2; k++)  y[k]=k+1;
rc=MPI_Send(&y, n2, MPI_INT, 0, tag2, comm);
cout <<"process "<< rank <<"  Send y = ";
for(int k=0; k<n2; k++) cout << y[k] <<"  ";
cout << endl;
}
else
{   cout<<"  process = "<< rank <<" no work !";   }
rc=MPI_Finalize();
return 0;   }
```

Задача 7
Процесс 0 передает вещественный массив x процессу size-1 и принимает от него целочисленный массив y; процесс size-1 принимает от процесса 0 массив x и передает массив y.
(совмещнная функции MPI_Sendrecv)

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Send_Recv
int main(int argc,char **argv)
{
int rank, size, rc, tag1=6, n1=5, n2=3, tag2=0;
float x[n1];  int y[n2];
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
if(rank==0)
{
for(int j=0; j<=10*(rank+1); j++)
  { for(int k=0; k<n1; k++)
      {  x[k]=(float)rand()/RAND_MAX;  }   }
rc=MPI_Sendrecv(&x, n1, MPI_FLOAT, size-1, tag1,
                &y, n2, MPI_INT, size-1, tag2, comm, &status);
cout <<"process 0 Send x = ";
for(int k=0; k<n1; k++) cout << x[k]<<"  ";   cout << endl;
cout <<"process 0 Recv y = ";
for(int k=0; k<n2; k++) cout << y[k] <<"  ";   cout << endl;
}
else if(rank==size-1)
{
for(int k=0; k<n2; k++)  y[k]=k+1;
rc=MPI_Sendrecv(&y, n2, MPI_INT, 0, tag2,
                &x, n1, MPI_FLOAT, 0, tag1, comm, &status);
cout <<"process "<< rank <<"  Send y = ";
for(int k=0; k<n2; k++) cout << y[k] <<"  ";    cout << endl;
cout <<"process "<< rank <<"  Recv x = ";
for(int k=0; k<n1; k++) cout << x[k] <<"  ";    cout << endl;
}
else
{   cout<<"  process = "<< rank <<" no work !";   }
rc=MPI_Finalize();
return 0;  }
```

Задача 8
Процесс 0 передает вещ. массив **x** процессу 1 и принимает от него массив **x**;
(функция замещающего обмена Sendrecv_replace)

```
#include <stdlib.h>
#include <mpi.h>
#include <iostream>
using namespace std;
// function Sendrecv_replace
// only 2 process !
int main(int argc,char **argv)
{
int rank, size, rc, tag1=6, n1=5, proc;
float x[n1];
MPI_Comm comm;
MPI_Status status;
rc=MPI_Init(&argc, &argv);
comm=MPI_COMM_WORLD;
rc=MPI_Comm_size(comm, &size);
rc=MPI_Comm_rank(comm, &rank);
cout <<"size = "<< size << endl;
for(int j=0; j<=10*(rank+1); j++)
{  for(int k=0; k<n1; k++)
   {   x[k]=(float)rand()/RAND_MAX; }   }
cout <<"process "<< rank <<"  Send x = ";
for(int k=0; k<n1; k++) cout << x[k] <<"  ";
cout << endl;
if(rank==0)  proc=size-1;
else if(rank==size-1)  proc=0;
else   proc=MPI_PROC_NULL;
rc=MPI_Sendrecv_replace(&x, n1, MPI_FLOAT, proc, tag1,
                                          proc, tag1, comm, &status);
cout <<"process "<< rank <<"  Recv x = ";
for(int k=0; k<n1; k++) cout << x[k] <<"  ";
cout << endl;
rc=MPI_Finalize();
return 0;     }
```