```
program hello
        use mpi
        implicit none
        integer::world_rank,world_size,source,dest,tag,ierr
        integer::send_data,recv_data,final_data
        integer::status(MPI_STATUS_SIZE)
        source = 0
        dest = 1
        tag = 0

        CALL MPI_Init(ierr)
        CALL MPI_Comm_size(MPI_COMM_WORLD,world_size,ierr)
        CALL MPI_Comm_rank(MPI_COMM_WORLD,world_rank,ierr)

        if(world_size < 3) then
                if(world_rank == source) then
                        print*,"need at least two process"
                end if
        CALL MPI_Abort(MPI_COMM_WORLD,1,ierr)
        end if

        if(world_rank == source) then
        send_data =  100
        CALL MPI_Send(send_data,1,MPI_INTEGER,dest,tag,MPI_COMM_WORLD,ierr)
                print*,"massage send:",send_data
        end if


        if(world_rank == dest)then
```

```
        CALL MPI_Recv(recv_data,1,MPI_INTEGER,source,tag,MPI_COMM_WORLD,status,ierr)
                print*,"massage recv:",recv_data
                final_data = 100
                final_data = final_data + recv_data
                print*,"final data:",final_data
        end if
        CALL MPI_Finalize(ierr)
end program hello
```

```
program marks

        implicit none

        character(len=100)::name

        integer::math,science,english
        integer::sum
        integer::avarage
        integer::unit_num

        unit_num = 10

        print*,"enter your name"
        read*,name
        print*,"enter maths marks"
        read*,math
        print*,"enter maths marks"
        read*,science
```

```
print*,"enter maths marks"
read*,english

sum = math + science + english
avarage =  sum/3
open(unit=unit_num,file="out.txt",status="replace")
select  case(avarage)

        case(75:100)
                print*,"A pass"
                write(unit_num,'(A)') trim(name)
                write(unit_num,'(A)')"A pass"

        case(65:74)
                print*,"B pass"
                write(unit_num,'(A)') trim(name)
                write(unit_num,'(A)')"B pass"

        case(55:64)
                print*,"C pass"
                write(unit_num,'(A)') trim(name)
                write(unit_num,'(A)')"C pass"

        case(35:54)
                print*,"S pass"
                write(unit_num,'(A)') trim(name)
                write(unit_num,'(A)')"S pass"

        case(0:34)
```

```
                print*,"F pass"
                write(unit_num,'(A)') trim(name)
                write(unit_num,'(A)')"you are fail"

        end select
        close(unit_num)
end program marks
```

```
#include<mpi.h>
#include<stdio.h>
int main(int argc,char** argv){
        MPI_Init(&argc,&argv);
        int size;
        int rank;
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        if(size< 3){
                if(rank == 0){
                        printf("neeed 3 ");
                }
                MPI_Abort(MPI_COMM_WORLD,1);
        }

        int num1 = 0;
        int num2 = 0;
        int ma=0;
        int fac=1;

 MPI_Barrier(MPI_COMM_WORLD);
```

```
        if(rank == 0){
                num1 = 2;
                ma+=num1;
                MPI_Send(&ma,1,MPI_INT,1,0,MPI_COMM_WORLD);
                printf("process 1: send %d\n",ma);
        }
        if(rank == 1){

                MPI_Recv(&ma,1,MPI_INT,0,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
                printf("process 2:num1 recv.");
                num2=3;
                ma+=num2;
                MPI_Send(&ma,1,MPI_INT,2,0,MPI_COMM_WORLD);
                printf("process 2: send %d\n",ma);
        }
        if(rank == 2){
                MPI_Recv(&ma,1,MPI_INT,1,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
                printf("process 3:num1 recv.%d\n",ma);
                for (int i = 1 ; i<=ma;i++)
                {
                        fac = fac * i;
                }
                printf("facto is %d\n",fac);
        }
        MPI_Finalize();
}
```