Ejercicio 1

Lo primero que se comprobara sera mirar si se es el proceso con id 0 o 1. En caso de ser 1 enviara un 1 y en caso de ser 0 se enviara un 0. En ambos casos enviaremos a el proceso con id + 2 comprobando anteriormente que sea menor al numero total de procesos.

Si el proceso no es ni 0 ni 1 recibiremos del proceso con id - 2 y enviaremos el mismo valor que recibimos a id + 2 si no es menor que el numero total de procesos.

```
if (rank == 0)
{
    value = 0;
    if (rank + 2 < size)
        MPI_Send(&value, 1, MPI_INT, rank + 2, 0, MPI_COMM_WORLD);
}
else if (rank == 1)
{
    value = 1;
    if (rank + 2 < size)
        MPI_Send(&value, 1, MPI_INT, rank + 2, 0, MPI_COMM_WORLD);
}
else
{
    MPI_Recv(&value, 1, MPI_INT, rank - 2, 0, MPI_COMM_WORLD, &status);
    if (rank + 2 < size)
        MPI_Send(&value, 1, MPI_INT, rank + 2, 0, MPI_COMM_WORLD);
}
```

Ejercicio 2

En este caso solo tendremos que modificar el bucle que realiza la distribución. Para esto calcularemos donde empezara cada proceso multiplicando el id del proceso por el numero de cálculos que realizara cada procesador y se le sumara uno.

Para calcular cuando termina simplemente le sumaremos a la posición de inicio el numero de cálculos que realiza cada procesador. El numero de cálculos de cada proceso se obtendrá dividiendo el total de pasadas por el numero de procesos que ejecutamos. En el caso del ultimo proceso tendremos que comprobar que el final sea menor que el total de pasadas que queremos realizar.

```
int numRepes = ceil((float)n/numprocs);
int inicio = numRepes * myid + 1;
int final = inicio + numRepes-1;
final = final < n ? final : n;
sum = 0.0;
```

```
    for (i = inicio; i <= final ; i++) { x="h" * ((double)i - 0.5); sum
+="4.0" (1.0 x); } < pre>
```

Ejercicio 3

En el ejercicio tres tenemos que realizar la misma división que en el caso anterior para rellenar el vector B. Lo único que cambia es que en este caso el total siempre un múltiplo del numero de procesos que tenemos ejecutando. Por tanto no tendremos que mirar si el ultimo procesador realiza mas ejecuciones de las que debería.

```
    long numRepes = ceil((float)tama/size);
    long inicio = numRepes * rank ;
    long final = inicio + numRepes;

    //printf("%d:%ld:%ld\n",size,inicio,final);
    for (long i = inicio; i < final ; i++)
    {
        VectorB[i] = (i + 1)*10;
    }
```

También tendremos que cambiar la multiplicación. Tendremos que utilizar la parte del vector B que pertenece a ese proceso. Para esto solo tendremos que iniciar sumándole a i el inicio de nuestro proceso.

```
    long producto = 0;
    for (long i = 0; i < tama / size; ++i) {
        producto += VectorALocal[i] * VectorB[i + inicio];
    }
```

Ejercicio 4

En este ejercicio tendremos que comprobar que el proceso tiene un color igual a 1 , esto indica que es impar, y que tiene es el proceso 0 de los impares. Con esto obtendremos el proceso con id 1 que inicializara el vector que pasaremos con el scatter. Para esto utilizaremos la variable color y la variable rank_nuevo que indicara el id del proceso en el nuevo comunicador.

A continuación realizaremos el scatter para todos los proceso que tengan el color igual a 1, afectando así solo a los impares. En el scatter utilizaremos el nuevo comunicador comm y repartiremos un entero a cada proceso.

```
   if (color == 1 && rank_nuevo == 0)
    {
        Vector.resize(size_nuevo, 0);
        for (int i = 0; i < size_nuevo; i++)
```

```
            Vector[i] = i;
    }
    if (color == 1){


        MPI_Scatter(&Vector[0], // Valores a compartir
                    1,          // Cantidad que se envia a cada proceso
                    MPI_INT,    // Tipo del dato que se enviara
                    &recibido,  // Variable donde recibir los datos
                    1,          // Cantidad que recibe cada proceso
                    MPI_INT,    // Tipo del dato que se recibira
                    0,          // proceso principal que reparte los datos
                    comm);      // Comunicador (En este caso, el global)
                    printf("Proceso %d de los impares y me han repartido el
valor %d\n", rank_nuevo, recibido);
    }
```