Задача 1. Да се напише програма во која ќе се иницијализира комуникација помеѓу два процеси, така што кај еден од двата процеси од тастатура ќе се вчита цел број и истиот ќе се прати на вториот процес. По приемот, вториот процес треба да ја отпечати добиената вредност.

Задача 2. Да се напише програма во која ќе се иницијализира комуникација помеѓу два процеси, така што кај едниот од двата процеси од тастатура ќе се вчита димензија на низа и елементите на низата, по што низата ќе се прати на вториот процес. По приемот, вториот процес треба да ја отпечати добиената вредност.

Напомена: За комуникацијата да се користат MPI_Send и MPI_Recv функциите.

Задача 3. Да се напише програма во која ќе се иницијализира комуникација помеѓу два процеси, така што кај едниот од двата процеси од тастатура ќе се вчита димензија на низа и елементите на низата, по што низата ќе се прати на вториот процес. По приемот, вториот процес треба да ја отпечати добиената вредност.

Напомена: За комуникацијата да се користат MPI_Send и MPI_Recv функциите. Меѓутоа, за разлика од решението на претходната задача, комуникацијата да се реализира само со еден пар MPI_Send и MPI_Recv повици.

Задача 4. Да се напише програма во која во нулти процес од тастатура ќе се вчита димензија на низа, по што низата ќе се пополни со случајно генерирани вредности во опсег од 0 до 50. Потоа, нултиот процес низата треба да ја прати до сите останати процеси.

Напомена: За комуникацијата да се користат MPI_Send и MPI_Recv функциите.

Задача 5. Да се напише програма во која во нулти процес од тастатура ќе се вчита димензија на низа, по што низата ќе се пополни со случајно генерирани вредности во опсег од 0 до 50. Потоа, нултиот процес низата треба да ја прати до сите останати процеси.

Напомена: За комуникацијата да се користи MPI_Bcast функцијата.

Задача 6. Да се напише програма во која ќе се направи споредба на брзината на извршување на MPI_Send и MPI_Recv комуникација наспроти MPI_Bcast комуникацијата. Споредбата да се прави за праќање на низа од изворен процес кон сите останати процеси.

Напомена: За мерење на времето да се користи MPI_Wtime функцијата.

Задача 7. Да се напише програма во која ќе се разгледаат сите можни влезни комбинации за логичкото коло прикажано на сликата со цел да се пронајдат и отпечатат оние комбинации за кои

на излез од колото се добива вредност 1. Истовремено, секој процес треба да чува информација за тоа колку од разгледаните комбинации го задоволуваат условот. На крај, користејќи ги локалните информации кај секој од процесите во главниот процес да се пресмета и отпечати бројот на комбинации кои го решаваат колото. Задачата да се реши со циклична распределба.

Задача 8. Да се напише програма во која ќе се разгледаат сите можни влезни комбинации за логичкото коло прикажано на сликата со цел да се пронајдат и отпечатат оние комбинации за кои на излез од колото се добива вредност 1. Истовремено, секој процес треба да чува информација за тоа колку од разгледаните комбинации го задоволуваат условот. На крај, користејќи ги локалните информации кај секој од процесите во главниот процес да се пресмета и отпечати бројот на комбинации кои го решаваат колото. Задачата да се реши со сегментација на интервалот.

Задача 9. Да се напише програма во која ќе се илустрираат разликите помеѓу блокирачки и неблокирачки повици. За секој пар повици да се разгледа статусната променлива од комуникацијата.

Задача 10. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде минималниот елемент, а работата да се подели на сите процеси рамномерно.

Напомена: Праќањето да се направи со MPI_Bcast функција.

Задача 11. Да се напише програма во која во главната програма се внесуваат елементи на низа. Од внесената низа да се најде минималниот елемент и процесот во кој е најден минимумот, а работата да се подели на сите процеси рамномерно.

Напомена: За собирање на информациите да се користи MPI_Reduce.

Задача 12. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде минималниот елемент и индексот на кој е најден минимумот, а работата да се подели на сите процеси рамномерно.

Напомена: За собирање на информациите да се користи MPI_Reduce.

Задача 13. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде минималниот елемент во низата, а работата да се подели на сите процеси рамномерно.

Напомена: Секој процес да го добие само својот дел од низата. Да се користи MPI_Scatter.

Задача 14. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде минималниот елемент и индексот на кој е најден минимумот, а работата да се подели на сите процеси рамномерно.

Напомена: Секој процес да го добие само својот дел од низата. Да се користи MPI_Scatter. Ако бројот на елементи не е делив со бројот на процеси да се затвори паралелниот регион.

Задача 15. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде минималниот елемент и индексот на кој е најден минимумот, а работата да се подели на сите процеси рамномерно.

Напомена: Секој процес да го добие само својот дел од низата. Да се користи MPI_Scatterv.

Задача 16. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде колку елементи се парни и таа информација да се прати до нултиот процес. На крај, во нултиот процес да се отпечати колку елементи вкупно го задоволуваат условот.

Напомена: Секој процес да го добие само својот дел од низата. Да се користи MPI_Scatterv.

Задача 17. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде колку елементи се парни и таа информација да се прати до нултиот процес. На крај, во нултиот процес да се отпечати колку елементи што го задоволуваат условот биле најдени кај секој процес соодветно.

Напомена: Секој процес да го добие само својот дел од низата. Да се користи MPI_Scatterv.

Задача 18. Да се напише програма во која во главната функција се внесуваат елементи на низа. Од внесената низа да се најде колку елементи се парни и таа информација да се прати до нултиот процес. На крај, во нултиот процес да се отпечати колку елементи што го задоволуваат условот биле најдени кај секој процес соодветно. Дополнително да се пратат и да се отпечатат и самите елементи кои го задоволуваат условот.

Напомена: Секој процес да го добие само својот дел од низата. Да се користи MPI_Scatterv.
Можно е секој процес да врати различен број на елементи, па да се користи MPI_Gatherv.

Задача 19. Да се напише програма во која во ќе се креира нов податочен тип за MPI и истиот ќе се искористи за MPI_Send/MPI_Recv комуникација помеѓу два процеси.

Задача 20. Да се напише програма во која секој процес ќе прати и прими низи со различна димензија од секој останат процес.

Напомена: Да се користи MPI_Alltoall и MPI_Alltoallv.

Задача 21. Да се напише програма во која ќе се помножат две матрици, при што матриците да се дефинираат со статички димензии. Секој процес да го земе само својот дел од работата. По пресметка, сегментите од локалните пресметки да се вратат во главниот процес како новодобиена матрица и истата да се отпечати на екран.

Задача 22. Да се напише програма во која ќе се алоцира матрица чии димензии ќе се вчитаат од тастатура. Кај матрицата треба да се пронајдат и отпечатат оние елементи кои се парни. Притоа, секој процес треба да го земе само својот дел од работата.

Функции:

int MPI_Send (void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

int MPI_Recv (void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)

int MPI_Get_count (MPI_Status *status, MPI_Datatype datatype, int *count)

int MPI_Probe (int source, int tag, MPI_Comm comm, MPI_Status *status)

int MPI_Isend (void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)

int MPI_Irecv (void *buf, int count, MPI_Datatype, int source, int tag, MPI_Comm comm, MPI_Request *request)

int MPI_Wait (MPI_Request *request, MPI_Status *status)

int MPI_Bcast (void *buf, int count, MPI_Datatype datatype, int root, MPI_Comm comm)

int MPI_Gather (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

int MPI_Gatherv (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, const int *recvcounts, const int *displs, MPI_Datatype recvtype, int root, MPI_Comm comm)

int MPI_Scatter (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

int MPI_Scatterv (void *sendbuf, const int *sendcounts, const int *displs, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)

```
int MPI_Reduce (void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op
op, int root, MPI_Comm comm)
```