ЗАДАЧИ К ЭКЗАМЕНУ
Дисциплина «Параллельное программирование».

1. Многопоточная программа «Test». Напишите OpenMP-программу, в которой создается 4 нити и каждая нить выводит на экран строку «Test».

2. Напишите OpenMP-программу, в которой создается k нитей, и каждая нить выводит на экран свой номер и общее количество нитей в параллельной области в формате:
I am <Номер нити> thread from <Количество нитей> threads!

3. Общие и частные переменные в OpenMP: программа «Скрытая ошибка»
Работа с данными shared и private. Напишите OpenMP-программу, в которой создается k нитей, и каждая нить выводит на экран свой номер через переменную rank следующим образом:
rank = omp_get_thread_num();
printf("I am %d thread.\n", rank);

4. Общие и частные переменные в OpenMP: параметр reduction
Напишите OpenMP-программу, в которой две нити параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью оператора if языка C. Для сложения результатов вычисления нитей воспользуйтесь OpenMP-параметром reduction.

5. Напишите OpenMP-программу, в которой k нитей параллельно вычисляют сумму чисел от 1 до N. Распределите работу по нитям с помощью OpenMP-директивы for.

6. Директивы создания параллельных секций sections и section. Напишите программу, содержащую 3 параллельные секции, внутри каждой из которых должно выводиться сообщение:
[<Номер нити>]: came in section <Номер секции>

7. Напишите OpenMP программу, в которой параллельно вычисляется сумма чисел от 1 до N. Используйте директиву atomic.

8. Напишите MPI программу, в которой каждый процесс выводит на экран свой номер и общее количество процессов в приложении в формате:
I am <Номер процесса> process from <Количество процессов> processes!

9. Напишите программу, в которой каждый процесс с четным номером выводит на экран строку « I am <Номер процесса>: FIRST! », а каждый процесс с нечетным номером – « I am <Номер процесса>: SECOND! ». Процесс с номером 0 должен вывести на экран общее количество процессов в приложении в формате « <Количество процессов> processes. ».

10. MPI-функции блокирующей передачи сообщений точка-точка MPI_Send и MPI_Recv. Напишите MPI-программу, в которой с помощью данных функций процесс с номером 0 отправляет сообщение процессу с номером 1. Процесс 1 выводит полученное сообщение на экран.

11. MPI-функция широковещательной рассылки данных MPI_Bcast. Напишите MPI-программу, которая в строке длины n определяет количество вхождений символов. Ввод

данных должен осуществляться процессом с номером 0. Для рассылки строки поиска и ее длины по процессам используйте функцию MPI_Bcast.

12. Напишите MPI программу, в которой производится широковещательная рассылка сообщения с помощью функции MPI_Bcast, но только по процессам с четным номером. Для рассылки сообщения создайте новый коммуникатор. Каждый процесс приложения должен выводить на экран « MPI_COMM_WORLD:
<номер процесса в коммуникаторе MPI_COMM_WORLD> from <количество процессов в коммуникаторе MPI_COMM_WORLD>. New comm: <номер процесса в новом коммуникаторе> from <количество процессов в новом коммуникаторе>. Message = <сообщение> »

13. Напишите программу, в которой нулевым процессом динамически запускается еще n процессов. Каждый процесс в программе выводит сообщение в формате « I am <Номер процесса> process from <Количество процессов> processes! My parent is <Номер процесса родителя> ».