Dated :
Assessment No. : 5

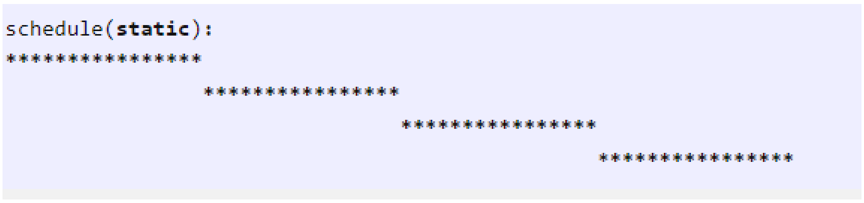
SCENARIO – I

Write a simple OpenMP program to employ a *'Scheduling'* clause to learn the use of static scheduling. In order to explore its practical use, you are advised to read and understand the following statements.

1. Parallelize for loop with 64 iterations
2. Use four threads to parallelize the for loops
3. Each row of stars in the examples represents a thread and each column represents an iteration

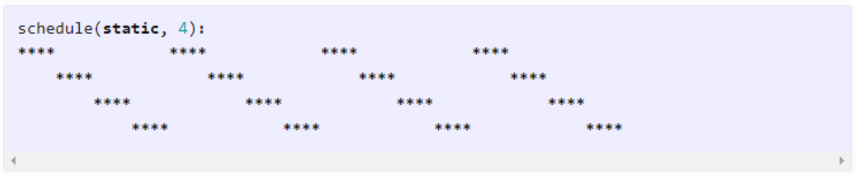
Execution Scenario:

The first scenario (schedule (static)) has 16 stars in the first row. This means that the first tread executes iterations 1, 2, 3, …, 15 and 16. The second row has 16 blanks and then 16 stars. This means that the second thread executes iterations 17, 18, 19, …, 31, 32. Similar applies to the threads three and four.

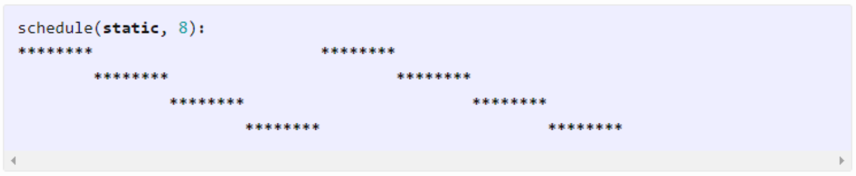


```
schedule(static):
****************
                ****************
                                ****************
                                                ****************
```

Now, We see that for schedule(static) OpenMP divides iterations into four chunks of size 16 and it distributes them to four threads. For schedule (static, 4) and schedule (static, 8) OpenMP divides iterations into chunks of size 4 and 8, respectively. An illustration is as follows:



```
schedule(static, 4):
****            ****            ****            ****
    ****            ****            ****            ****
        ****            ****            ****            ****
            ****            ****            ****            ****
```



```
schedule(static, 8):
********                        ********
        ********                        ********
                ********                        ********
                        ********                        ********
```

BRIEF ABOUT YOUR APPROACH:

SOURCE CODE:

EXECUTION:

Dated :
Assessment No. : 5

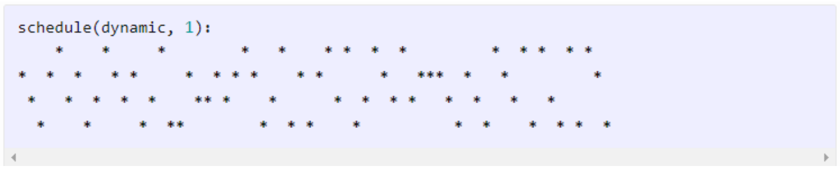
RESULTS:

SCENARIO – II

Write an OpenMP program to specify that the schedule(dynamic, chunk-size) clause of the loop construct specifies that the for loop has the dynamic scheduling type.

Description

There is no particular order in which the chunks are distributed to the threads. The order changes each time when we execute the for loop. If we do not specify chunk-size, it defaults to one.

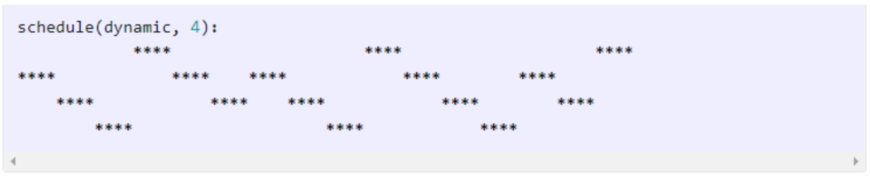


```
schedule(dynamic, 1):
    *    *     *        *   *    * *  *  *         *  * *  * *
*  *  *   * *     *  * * *    * *      *   ***  *   *         *
 *   *  *  *  *    ** *    *      *  *  * *   *  *   *   *
  *    *     *  **        *  * *    *          *  *    *  * *  *
```

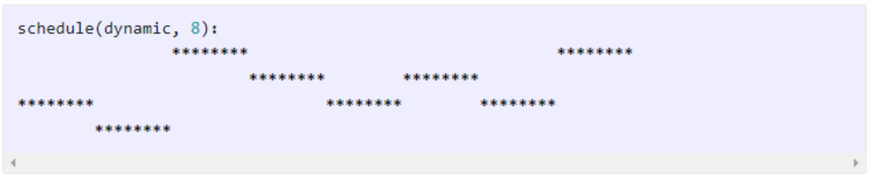
Execution Scenario

Show that the OpenMP schedule (dynamic, 4) and schedule (dynamic, 8) divides iterations into chunks of size four and eight, respectively. The distribution of chunks to the threads has no pattern.

Also, prove that the dynamic scheduling type is appropriate when the iterations require different computational costs. This means that the iterations are poorly balanced between each other. The dynamic scheduling type has higher overhead then the static scheduling type because it dynamically distributes the iterations during the runtime.



```
schedule(dynamic, 4):
            ****                    ****                    ****
****            ****    ****            ****        ****
    ****            ****    ****            ****        ****
        ****                    ****            ****
```



```
schedule(dynamic, 8):
                ********                                ********
                        ********        ********
********                        ********        ********
        ********
```

BRIEF ABOUT YOUR APPROACH:

SOURCE CODE:

EXECUTION:

RESULTS: