DEPARTMENT OF INFORMATION TECHNOLOGY, NITK, SURATHKAL

Parallel Computing

Lab-4-26th August 2024

Note: Execute the following programs and take screen shots of the output. Write the analysis of the results and upload the code, results and analysis as a single file in the Moodle.

Total marks: 3+3+2+2 = 10

1. To understand the concept of schedule. Write the observation using schedule (static, 5), schedule(dynamic,5) and schedule (guided,5) **[1+1+1=3]**

```
#include<stdio.h>
#include <omp.h>
#include <stdlib.h>
int main(void)
{
int i;
#pragma omp parallel num_threads(4)]
{
	#pragma omp for schedule(guided,5) private(i)
	for (i=0;i<25;i++)
	{
		printf("tid=%d, i=%d\n", omp_get_thread_num(),i);
	}
}
return 0;
}
```

2. Execute the following code to understand the concept of collapse(). Consider three for loops and check the result with **no collapse(), collapse(2),** and **collapse(3).**

Nocollapse(): **[1+1+1=3]**

```
#include<stdio.h>
#include <omp.h>
#include <stdlib.h>
```

```
int main(void)
{
int i,j,k;
#pragma omp parallel
{
        #pragma omp for schedule(static,3) private(i,j,k)
        for (i=0;i<4;i++)
          for (j=0;j<3;j++)
             for (k=0;k<2;k++)
             {
                int tid= omp_get_thread_num();
                printf("tid=%d i=%d j=%d k=%d\n",tid,i,j,k);
             }
   }
return 0;
}
```

collapse(2):

```
#include<stdio.h>
#include <omp.h>
#include <stdlib.h>

int main(void)
{
int i,j,k;
#pragma omp parallel
{
        #pragma omp for schedule(static,3) private(i,j,k)
collapse(2)
        for (i=0;i<4;i++)
          for (j=0;j<3;j++)
             for (k=0;k<2;k++)
             {
                int tid= omp_get_thread_num();
                printf("tid=%d i=%d j=%d k=%d\n",tid,i,j,k);
             }
   }
return 0;
}
```

collapse(3):

```
#include<stdio.h>
#include <omp.h>
#include <stdlib.h>

int main(void)
{
int i,j,k;
#pragma omp parallel
{
      #pragma omp for schedule(static,3) private(i,j,k) collapse(3)
      for (i=0;i<4;i++)
        for (j=0;j<3;j++)
          for (k=0;k<2;k++)
          {
            int tid= omp_get_thread_num();
            printf("tid=%d i=%d j=%d k=%d\n",tid,i,j,k);
          }
  }
return 0;
}
```

3. Execute the following code and observe the working of threadprivate directive and copyin clause.

[1+1=2]

```
#include<stdio.h>
#include <omp.h>
#include <stdlib.h>
int tid,x;
#pragma omp threadprivate(x,tid)
void main()
{
      x=10;
      #pragma omp parallel num_threads(4) copyin(x)
      {
      tid= omp_get_thread_num()
```

```
	#pragma omp master
	{
		printf("Parallel region 1 \n");
		x=x+1;
	}
#pragma omp barrier
if (tid==1)
x=x+2;
printf("thread %d value of x is %d\n", tid,x);
}
#pragma omp parallel num_threads(4)
{
	#pragma omp master
{
		printf("Parallel region 2 \n");
}
#pragma omp barrier
printf("thread %d value of x is %d\n", tid,x);
}
printf("value of x in main region is %d\n",x);
}
```

4. In the above program (Program No.3),

[1+1=2]

(i) remove the copyin clause and check the output.

```
#include<stdio.h>
#include <omp.h>
#include <stdlib.h>
int tid, x;
#pragma omp threadprivate(x,tid)
void main()
{
	x=10;
	#pragma omp parallel num_threads(4)
	{
	tid= omp_get_thread_num()
	#pragma omp master
	{
		printf("Parallel region 1 \n");
		x=x+1;
	}
```

```
#pragma omp barrier
if (tid==1)
x=x+2;
printf("thread %d value of x is %d\n", tid,x);
}
#pragma omp parallel num_threads(4)
{
	#pragma omp master
{
		printf("Parallel region 2 \n");
}
#pragma omp barrier
printf("thread %d value of x is %d\n", tid,x);
}
printf("value of x in main region is %d\n",x);
}
```

(ii) remove the copyin clause, initialize **x** globally and check the output.

```
 #include<stdio.h>
#include <omp.h>
#include <stdlib.h>

int tid, x=10;
#pragma omp threadprivate(x, tid)
void main()
{

	#pragma omp parallel num_threads(4)
	{
	tid= omp_get_thread_num()
	#pragma omp master
	{
		printf("Parallel region 1 \n");
		x=x+1;
	}
#pragma omp barrier
if (tid==1)
x=x+2;
printf("thread %d value of x is %d\n", tid,x);
```

```
}
#pragma omp parallel num_threads(4)
{
      #pragma omp master
{
            printf("Parallel region 2 \n");
}
#pragma omp barrier
printf("thread %d value of x is %d\n", tid,x);
}
printf("value of x in main region is %d\n",x);
}
```

****

1. write an OpenMP program to demonstrate sharing of section work by
2. performing arithmetic operations on one

dimensional array by thread