Ahmed Alaa

1) **OpenMP is a ..... level programming model which is ..... programming abstraction.**
   a) low, shared memory
   b) low, distributed memory
   c) high, shared memory
   d) high, distributed memory
2) **Multi-thread programs have .... entry point(s) and .... exit point(s).**
   a) single, single
   b) single, multiple
   c) multiple, single
   d) multiple, multiple
3) **In Java, a low-priority thread that runs in the background to perform tasks such as garbage collection is called ....**
   a) orphan Threads
   b) Daemon Threads.
   c) Zombie Threads
   d) Confused Threads
   e) Lonely Threads
4) **Directives are handled in ..... stage.**
   a) Prepossessing
   b) Compilation
   c) Assembling
   d) Linking
   e) Runtime

5) **In shared memory systems, any access from any processing element to the same address has equal latency** (……..…..)
6) **In general, Master thread must be the last thread to be terminated, however, in openMP, Master thread can be terminated before their user threads.** (……..…..)
7) **PThreads is a distributed memory system.** (……..…..)

Answers:

1) **c**
2) **b**
3) **b**
4) **a**
5) **T**
6) **F**
7) **F**

Ahmed Hosny

1) **Within a parallel region, declared variables are by default _____.**
   a) Private
   b) Loco
   c) Shared
   d) None of the other answers
   e) Local
2) **what is the library should be included to use open MP functions**
   a) #include<omp.h>
   b) #include<paralle.h>
   c) #include<openmp.h>
   d) #include<mp.h>
   e) #include<open.h>
3) **in the end of the parenthesis of**

```
#pragma omp parallel
{
}
```

   there is ……..

   a) implicit barrier
   b) implicit critical
   c) implicit atomic
   d) implicit Shared
   e) None of the other answers
4) **in the following code:**

```
int main()
{
        omp_set_num_threads(3);
        int id = omp_get_num_thread();
}
```

   the value of the id is:

   a) 3
   b) 2
   c) 1
   d) 0
   e) None of the other answers
5) **True or false:** Code in an OpenMP program that is not covered by a pragma is executed by all threads (……..)
6) **OpenMP program is an API for:**
   a) shared memory parallel programming
   b) distributed memory parallel programming
   c) Both of above
   d) None of above
7) **T/F : Code in an OpenMP program that is not covered by a pragma is executed by all threads.** (…….)

8) **in the following code:**

```
int main()
{
        omp_set_num_threads(3);
        int id = omp_get_thread_num();
}
```

the value of the id is:

a) 3

b) 2

c) 1

d) 0

e) None of the other answers

9) **in the following code:**

```
int main()
{
        int sum = 10;
        #pragma omp parallel
        {
                sum+=2;
        }
}
```

this code can cause:

a) false sharing
b) race condition
c) None of the other answers

10) **Directives appear just before a block of code, which is delimited by:**

a) (…)
b) […]
c) { … }
d) < … >

Answers:

1) d
2) a
3) a
4) c
5) F
6) a
7) F
8) d
9) b
10) c

abdo salm

1) **Which of the following is not considered work sharing construct?**
   a) Single
   b) Master
   c) Section
   d) Critical
   e) For
2) **there is implicit barrier at the end of master construct**. (……..)
3) **The following code will result in a data race:**

```
#pragma omp parallel for
 for (i=1; i < 10; i++)
 {
        factorial[i] = i * factorial[i-1];
 }
```

   a) True
   b) False
4) **Name at least one difference between master construct and single construct?**
5) **A _____ construct must be enclosed within a parallel region in order for the directive to execute in parallel.**
   a) Parallel sections
   b) Critical
   c) Single
   d) work-sharing
6) **The _____ specifies that the iterations of the for loop should be executed in parallel by multiple threads.**
   a) Sections construct
   b) for pragma
   c) Single construct
   d) Parallel for construct
7) **In OpenMP, assigning iterations to threads is called _____**
   a) Scheduling
   b) Static
   c) Dynamic
   d) Guided
8) **The master region can be executed by any thread including the master thread.** (……..)

Answers:

1) **d**
2) **F**
3) **T**
4) **In master construct, only the master (thread with ID = 0) executes that part of code and the rest will skip that part of code, but in case of single construct, only one thread can execute that part of code no matter what its ID is, in master construct there is no implicit barrier at the end of the construct , while in the single construct , there is implicit barrier at the end of the construct.**
5) **d**
6) **b**
7) **a**
8) **F**

Nour Ziad

1) **Variables: A=1 ; B=1 ; C=1**
   #pragma omp parallel private(B) firstprivate(C)
   Are A,B,C local to each thread or shared inside the parallel region?
   What are their initial values inside?
2) **T/F: if we declared any variable in the sequential part of the program then it can only be shared among all threads.** (……)
3) **T/F: If the data-sharing attribute of a variable is private within a construct, a separate local copy of the same variable is created for every thread (including the master thread).** (……)
4) …… **initializes each private copy with the corresponding value from the master thread.**
   a) Firstprivate
   b) lastprivate
   c) nowait
   d) Private (OpenMP) and reduction.
5) **A** …… **in OpenMP is just some text that modifies a directive.**
   a) data environment
   b) clause
   c) task
   d) Master thread
6)

```
int i;
double sum;
// sum = 1;
#pragma omp parallel for  reduction(* : sum)
 for (i=1; i <= 4; i++)
          sum = sum * i;
  printf("The sum is %lf \n",sum);
```

   Do we have to uncomment line 3 so that my code run correctly? give a reason.

7)

```
int i;
double sum = 0.0;
// sum = 1;
#pragma omp parallel for firstprivate(sum) num_threads(1)
 for (i=1; i <= 4; i++)
          sum = sum + 1;
  printf("The sum is %lf \n",sum);
```

   what is the value of (sum) after executing this code?

Answers:

1) **B & C are local to each thread**
   A is shared among threads
   Initial values : A=1 , B=garbage , C=1
2) **F**
3) **T**
4) **a**
5) **b**
6) **No, reduction already initializes variables in the list with the identity of the operator**
   Local copies of sum will be initialized with one.
7) **0.0**

Ahmed Madbouly

1) **the thread can change its own ID (THREAD_NUM) during execution.** (……..)
2) **can multiple threads have same ID (THREAD_NUM) in Nested parallelism.** (……..)
3) **the expected output if we call function omp_get_num_threads() in serial region is .....**
   a) runtime error
   b) compile error
   c) 1
   d) 0
4) **<if we disable the nested parallelism ,and using the following construct>**

```
#paragma omp parallel num_threads(3)
{
        //region 1
        #paragma omp parallel num_threads(4)
        {
                //region 2
        }
}
```

   1) **the number of threads working in region 1 (at the same time) is:**
      a) 1
      b) 2
      c) 3
      d) 4
   2) **the number of threads working in region 2 (at the same time) is:**
      a) 1
      b) 2
      c) 3
      d) 4
   3) **number of teams working in region 1 (at the same time) is**
      a) 1
      b) 2
      c) 3
      d) 4
   4) **number of teams working in region 2 (at the same time) is:**
      a) 1
      b) 2
      c) 3
      d) 4
   5) **we must initialize the Enviroment variables At the beginning of the program to use it .** (……..)
   6) **we could override the default value of the Environment variable (OMP_NUM_THREADS) inside the program.** (……..)
   7) **The default value of the environment variable (OMP_NUM_THREADS) is the number of processors in your machine.** (……..)
   8) **we couldn't enter the single region with more than one thread.** (……..)

Answers:

1) **T**
2) **T**
3) **c**
4) **1) c 2) c 3) a 4) c**
5) **F**
6) **T**
7) **T**
8) **F**

Mennatallah

8) **Which of the following decides when a task is executed?**
   a) runtime system
   b) programmer
   c) thread
9) **A thread generates a task when it encounters:**
   a) task construct
   b) parallel construct
   c) single construct

3) In the flowing 2 versions of a program to execute 2 tasks:

```
#pragma omp parallel
{
        #pragma omp single nowait
        {
                #pragma omp task
                b = beta();

                a = alpha();
        }
}
```

```
#pragma omp parallel
{
        #pragma omp single nowait
        {
                #pragma omp task
                b = beta();
                #pragma omp task
                a = alpha();
        }
}
```

a) Why in the second pragma, nowait is used ?
b) What is the difference between the 2 versions ?

4) **What does the nowait clause do?**
   a. Skips to the next OpenMP construct
   b. Prioritizes the following OpenMP construct
   c. Removes the synchronization barrier from the previous construct
   d. Removes the synchronization barrier for the current construct

Answers:

1) **a**
2) **a**
3) **a) To eliminate implicit barrier**
   b) is that the program on the left generates task beta() and immediately executes alpha() on the same thread, while the program on the right simply generates beta() and alpha() for execution by any thread

4) d