Please read these instructions carefully!

- Answer the test questions in the separate answer sheet.
- You may use the back of all the paper sheets as drafting area (*rascunho*).
- To replace an answer, draw a (well visible) cross over the canceled choice e and fill the circle of the new choice (✗ ○ ○ ● ○).
- To reactivate a canceled choice, draw a (well visible) cross over the new answer to be canceled and draw a (well visible) circle around the (preciously canceled) choice you want to reactivate. (● ○ ○ ✗ ○).
- This test has **??** QUESTIONS, each question valued as 20/**??** points.
- **POINTS LOST** for each wrong answer (in percentage of the question value): $1st = 0\%$, $2nd = 11,11\%$, $3rd = 22,22\%$, $4th+ = 33,33\%$.

Name: _____ Number: _____

---

1. Why do one uses parallel computing?
   A. To splitvery large data sets into smaller pieces to be processed independently.
   B. To minimize the latency on accessing data by creating and managing multiple replicas of the data.
   C. Solving larger problems in the same time.
   D. Merging different unrelated tasks into a single computation.

---

2. In the context of parallel computing, in which process the original program is decomposed into basic sub-program units or tasks?
   A. Cooperation. B. Scheduling. C. Partitioning. D. Synchronization.

---

3. Serialization is…
   A. The act of processing a large dataset as a sequential stream.
   B. The act of putting some set of operations into a specific concurrent order.
   C. The act sorting some set of operations into non-overlapping execution times.
   D. The act enforcing some set of operations to waiting for some other operations to conclude.

---

4. Point out the **FALSE** statement.
   A. The MapReduce framework is appropriate for processing large streams of data.
   B. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner.
   C. The MapReduce framework operates exclusively on pairs key/value.
   D. The local storage area in Mappers expire (becomes invalid) after each job is concluded.

---

5. Which of the following patterns is similar to map but directed to data streams?
   A. Stencil B. Split C. Scatter D. Farm

---

6. Which of the following statements better express the behaviour of the MAP parallel pattern?
   A. `REPEAT w=generate_data(); UNTIL w=false`
   B. `WHILE w=getWork() DO process(w); DONE`
   C. `FOREACH w IN foo DO process(w); DONE`
   D. `LOOP w=getWork(); process(w); FOREVER.`

---

7. Which parallel pattern **IS IMPLEMENTED** by the the OpenMP code block on the right?

```
void pattern(int n, double a[], double b[], double c[]) {
    #pragma parallel for
    for (i = 0;  i < n;  i++)
        a[i] = f(b[i], c[i]);
}
```

   A. Stencil. B. Reduce. C. Pack. D. Map.

---

8. Which of the following phrases about OpenMP is **TRUE**? *The statement following a* `#pragma omp single` *used inside a* `#pragma omp parallel` *will be executed by…*

   A. … one and only one thread.

   B. … at most one thread.

   C. … as many thread as there are in the `parallel` block.

   D. … at least one thread.

---

9. When a pthread starts by executing a function F, what happens to the stack frame (and to the local variables in that function) when the thread dies?

   A. The stack frame of F remains and is valid until the corresponding `pthread_join()`, when it is finally pop'ed.

   B. The stack frame of F is pop'ed and the variables cease to exist (are no longer valid).

   C. The stack frame of F is pop'ed but the variables remain valid until the corresponding `pthread_join()`.

   D. The values of the variables are kept until F is started again by another invocation of `pthread_create()`.

---

10. The first Lab assignment was to implement a parallel algorithm to approximate the value of π by using the Monte Carlo computation. Select the statement that broadly defines a Monte Carlo computation.

    A. A computation that rely on repeated random sampling to obtain numerical results.

    B. A computation that obtains numerical results by systematic approximating the target value by reaching a local maximum, and than introduce some noise to escape that local maximum, and keep on approximating the target value.

    C. A computation that approximates a value by using genetic algorithms.

    D. A computation involving the use of random numbers and floats/doubles.

---

11. Which types of data dependences **DO NOT** affect the correctness of the program?

    A. All except Read-after-Write and Write-after-Read.

    B. All except Read-after-Write, Write-after-Read and Write-after-Write.

    C. All except Read-after-Read and Write-after-Write.

    D. All do.

---

12. Identify the statement that is **TRUE** concerning the dependences that can be found in the following code block?

```
for (int i=1;  i < n;  i++)
  for (int j=0;  j < m;  j++)
    a[i][2*j] = a[i-1][2*j+1];
```

    A. The is a loop-carried output dependence on `a[i,j]`.

    B. There are a loop-carried anti-dependence on `i` and a loop-carried flow dependence on `j`.

    C. There are a loop-carried flow dependence on `i` and a loop-carried anti-dependence on `j`.

    D. There are no dependences (the statement is loop-independent on both `i` and `j`).

---

13. Which of the following strategies enables a larger speedup for a program that was initially sequential?

    A. Make 50% of the code 100 times faster?

    B. Make 90% of the code 3 times faster?

    C. Make 60% of the code 50 times faster?

    D. Make 70% of the code 30 times faster?

---

14. Which of the following expressions defines the cost of a parallel computation?
    n = #processors; $C(n)$ = cost; $S(n)$ = speedup; T_n = parallel execution time; T_s = sequential execution time; f = non-parallelizable fraction of the original program.

    A. $C(n) = T_n \times n$ B. $C(n) = (T_n \times f) \times n$ C. $C(n) = S(n)/n$ D. $C(n) = (f \times n)/T_n$

15. Given a parallel computation represented as a DAG (Direct Acyclic Graph) G, with work W and span S and P processes, which of the following frases is **FALSE**?

A. Each vertex in G is executed exactly once.

B. If a vertex u is ordered before vertex v in G, then v is not executed at a time step before u.

C. Every execution schedule has length at least $\frac{S}{P}$.

D. We define the *Span* as the length of the longest path in the dag.

---

16. In the Work-Span model, a critical path in the DAG (Direct Acyclic Graph) is:

A. A path that leads to a node with no outgoing edge.

B. A path that forms a cycle.

C. A path that includes the node with the higher number of incoming edges.

D. The path that takes longer to execute.

---

17. Select the situation that always requires synchronization between the processes.

A. Processes A releases a block of memory of size S_a and process B requests a block of memory of size $S_b < S_a$.

B. Process A releases a *lock* and process B acquires that *lock*.

C. Processes A and B read the same memory location.

D. Process B is a replica of process A and they are both executing the same computation.

---

18. Which of the following invariants do not apply to a initially empty bounded queue?

p = number of data items produced so far n = number of elements currently in the queue
c = number of data items consumed so far k = size of the queue

A. $(c \geq 0) \wedge (p \geq c) \wedge (p \leq c + k)$ B. $n = p - c$ C. $c < n + k$ D. $(n > 0) \Rightarrow (p > c)$

---

19. Given an atomic register R and let $\mathrm{T}_b(\mathrm{op})$ and $\mathrm{T}_e(\mathrm{op})$ be respectively the invocation and return of *op*, identify which of the following phrases is **FALSE**.

A. For any two operation invocations $op1$ and $op2$, $\mathrm{T}(op1) \neq \mathrm{T}(op2) \Rightarrow op1 \neq op2$.

B. An atomic register R can be accessed by two base operations: `R.read()` and `R.write(v)`.

C. Each invocation *op* of a read or write operation on an atomic shared register appears as if it was executed at a single point $\mathrm{T}(op)$ of the time line, where $\mathrm{T}_b(op) \leq \mathrm{T}(op) \leq \mathrm{T}_e(op)$.

D. For any two operation invocations $op1$ and $op2$, $\mathrm{T}_e(op1) > \mathrm{T}_e(op2) \Rightarrow \mathrm{T}(op1) > \mathrm{T}(op2)$.

---

20. Given the following implementation of a mutex (i is the process acquiring/releasing the mutex, j is the other process), indicate which of the phrases below is **TRUE**:

```
operation mutex_acquire(i) is
  do
    FLAG[i] = up;
    if (FLAG[j] == up) FLAG[i] = down;
  until (FLAG[i] == up);
end
```

```
operation mutex_release(i) is
    FLAG[i] = down;
end
```

A. The given implementation ensures both mutual exclusion and progress.

B. The given implementation does not ensure mutual exclusion neither progress.

C. The given implementation ensures progress but not mutual exclusion.

D. The given implementation ensures mutual exclusion but not progress.

---

21. Which of the following defines the **lock-freedom** progress property?

A. Even in the presence of contention, all threads will complete its operation in a bounded number of steps.

B. In the absence of contention, a thread never bars (blocks) the progress of any other thread.

C. In the presence of contention, at least one thread will complete its operation in a bounded number of steps.

D. In the absence of contention, at least one thread will complete its operation in a bounded number of steps.

22. Consider the implementation of the operation *remove* in a *lazy-list* as presented on the right. Which code is missing in line 6?

A. `while (current.key < key)`

B. `if (current.key < key)`

C. `if (current.key <= key)`

D. `while (current.key <= key)`

23. Consider the implementation of the operation *remove* in a *lazy-list* as presented on the right. Which code is missing in line 13?

A. `if (validate(pred.key, curr.key))`

B. `if (validate(pred, curr))`

C. `if (validate(pred.next, curr))`

D. `if (validate(pred.next, curr.prev))`

24. Consider the implementation of the operation *contains* in a *lazy-list* as presented on the right. Which code is missing in line 72?

A. `cuur.key == key && !curr.marked`

B. `cuur.key != key && !curr.marked`

C. `cuur.key != key || !curr.marked`

D. `cuur.key == key || !curr.marked`

```
1   public boolean remove(T item) {
2     int key = item.hashCode();
3     while (true) {
4      Node pred = head;
5      Node curr = head.next;
6      _____ {
7       pred = curr; curr = curr.next;
8      }
9      pred.lock();
10     try {
11       curr.lock();
12       try {
13         _____ {
14           if (curr.key != key) {
15             return false;
16           } else {
17             curr.marked = true;
18             pred.next = curr.next;
19             return true;
20           }
21         }
22       } finally {
23         curr.unlock();
24       }
25     } finally {
26       pred.unlock();
27     }
28   }
29 }

67 public boolean contains(T item) {
68   int key = item.hashCode();
69   Node curr = head;
70   while (curr.key < key)
71     curr = curr.next;
72   return _____;
73 }
```

25. Which of the following statements is **FALSE**. In the definition of the lockset algorithm…

A. When a thread t accesses location x, the lockset algorithm does $LockSet(x) = LockSet(x) \cap LocksHeld(t)$.

B. When initializing, the lockset algorithm does $LockSet(x) = \emptyset$.

C. When initializing, the lockset algorithm does $LocksHeld(t) = \emptyset$.

D. When a thread t releases a lock l, the lockset algorithm does $LocksHeld(t) = LocksHeld(t) \setminus l$.

26. Given the views $V_1, \cdots, V_4$ on the right, and the threads T_1, T_2, T_3, where $T_i // T_j$ means that thread T_i executes concurrently with thread T_j, and $T_i \Leftarrow < V_i, ..., V_k >$ means thread T_i executed the views $< V_i, ..., V_k >$ in that order, which of the following executions generate a **high-level data race**?

$V_1 = \{A, B, C, D\}$
$V_2 = \{A, B\}$
$V_3 = \{A, B, D\}$
$V_4 = \{B, D\}$

A. $T_1 \Leftarrow < V_1, V_4 > \;\; // \;\; T_2 \Leftarrow < V_2, V_3 >$.

B. $T_1 \Leftarrow < V_1 > \;\; // \;\; T_2 \Leftarrow < V_2 > \;\; // \;\; T_3 \Leftarrow < V_2, V_3 >$.

C. $T_1 \Leftarrow < V_1, V_3 > \;\; // \;\; T_2 \Leftarrow < V_2, V_4 >$.

D. $T_1 \Leftarrow < V_1 > \;\; // \;\; T_2 \Leftarrow < V_2 > \;\; // \;\; T_3 \Leftarrow < V_4 >$.

27. Concerning the Banker's algorithm, we say a system is in a safe state if…

A. There exist a sequence $< P_1, P_2, ..., P_n >$ of ALL the processes in the system such that the resources that P_i may still request can be satisfied by the currently the available resources + the resources held by all the $P_j : j \neq i$.

B. There exist a sequence $< P_1, P_2, ..., P_n >$ of ALL the processes in the system such that the resources that P_i may still request can be satisfied by currently available resources + the resources held by all the $P_j : j \leq i$.

C. There exist a sequence $< P_1, P_2, ..., P_n >$ of ALL the processes in the system such that the resources that P_i may still request can be satisfied by the currently the available resources + the resources held by all the $P_j : j < i$.

D. There exist a sequence $< P_1, P_2, ..., P_j >$ of SOME of the processes in the system such that the resources that P_i may still request can be satisfied by currently available resources + the resources held by all the P_j, with $j \leq i$.

Total

A	B	C	D
3	17	16	12

Maximum

	A	B	C	D
P1	0	2	1	0
P2	1	6	5	2
P3	2	3	6	6
P4	0	6	5	2
P5	0	6	5	6

Allocated

	A	B	C	D
P1	0	1	1	0
P2	1	2	3	1
P3	1	3	6	5
P4	0	6	3	2
P5	0	0	1	4

Available

A	B	C	D
1	5	2	0

Figure 1: Banker's Algorithm data

28. In one of the lab classes you were given a working program written in Java that **used a single hash-map**. Select the statement that is **TRUE** with respect to that lab assignment.

    A. The (original) code given was operational for multithreaded execution because the hash-map was implemented using non-blocking (lock-free) techniques.

    B. The (original) code given was operational for multithreaded execution because all the public methods of the hash-map object had the `synchronized` attribute.

    C. The code had already some routines to verify the consistency of the data during/after the execution of the program.

    D. You were asked to use lock objects to introduce hand-over-hand synchronization at the level of the collision lists of the hash-map.