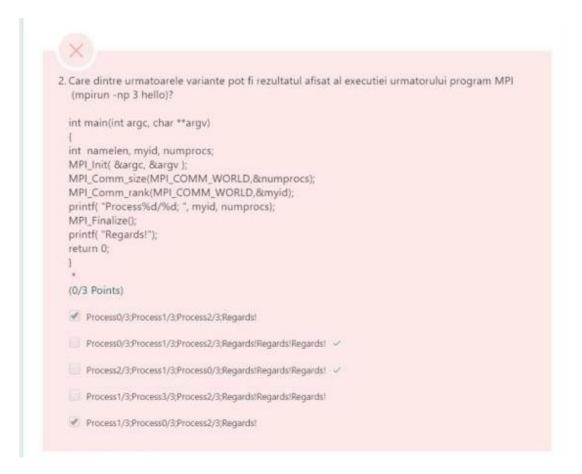
- 1. Scalabilitatea este mai mare pentru sistemele:
  - a. MPP
  - b. SMP
- 2. Latenta memoriei este.
  - a. rata de transfer a datelor din memorie catre processor
  - b. timpul in care o data ajunge sa fie disponibila la procesor dupa ce s-a initiat cererea.
- 3. Asigurarea cache coherency determina pentru scalabilitate o: Single choice.
  - a. Scadere
  - **b.** Crestere
  - **c.** Nu este legatura
- 4. Un calculator cu 1 procesor permite executie paralela? Required to answer. Single choice.
  - a. Da
  - b. Nu
- 5. "Context Switch" este mai costisitor pentru:
  - a. Procese
  - b. Threaduri
- 6. Thread1 executa {a=b+1; a=a+1} si Thread2 executa {b=b+1}. Apare "data race"? Required to answer. Single choice.
  - a. Da
  - b. Nu
- 7. Thread1 executa  $\{c=a+1\}$  si Thread2 executa  $\{b=b+a;\}$ . Apare "data race"?
  - a. Da
  - b. Nu
- 8. Intr-o executie determinista poate sa apara "race condition".
  - a. Fals
  - b. Adevarat
- 9. Granularitatea unei aplicatii paralele este
  - a. Definita ca dimensiunea minima a unei unitati secventiale dintr-un program, exprimata in numar de instructiuni
  - b. Este determinata de numarul de taskuri rezultate prin descompunerea calculului
  - c. Se poate aproxima ca fiind raportul din timpul total de calcul si timpul total de comunicare
- 10. Granularitatea unei aplicatii paralele este de dorit sa fie:
  - a. Mica
  - b. Mare
- 11. Granularitatea unui sistem paralel este de dorit sa fie:
  - a. Mica
  - b. Mare
- 12. Eficienta unui program parallel care face suma a 2 vectori de dimensiune n folosind p procese este:
  - a. Maxim 1

- b. Minim 1
- c. Egala cu p
- 13. Costul unei aplicatii paralele este optim daca
  - a.  $C=O(T_s*log p)$
  - b.  $C=O(T_s)$
  - c. C=Omega(T\_s)
- 14. Un semafor care stocheaza procesele care asteapta intr-o multime, se numeste:
  - a. Strong Semaphone (semafor puternic)
  - b. Weak Semaphor (semafor slab)
  - c. Semafor binar
- 15. Livelock descrie situatia inn care:
  - a. Un grup de procese/threaduri nu progreseaza datorita faptului ca isi cedeaza reciproc executia
  - b. Un grup de procese/threaduri nu progreseaza datorita faptului ca isi blocheaza reciproc executia
  - c. Un grup de procese/threaduri nu progreseaza datorita faptului ca nu se termina niciunul
- 16. Fata de Monitor, Semaforul este o structura de sincronizare:
  - a. De nivel inalt
  - b. De nivel jos
  - c. De acelasi nivel



```
1. Este posibil ca urmatorul cod MPI sa produca deadlock?
              int main(int argc,char *argv[]) {
              // var declaration... init....
              if (rank == 0) {
                 dest = source = 1;

rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);

rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &:Stat);
                  dest = source = 0;
                  rc = MPI_Recv(&immsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD, &Stat);
rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
              // ... finalizare
              (3/3 Points)
              DA
              · Nu ·
#pragma omp parallel shared(chunk,a) private(i,tid) num_threads(
       tid = omp_get_thread_num();
       indx = indx + chunk * tid:
       for (i = indx; i < indx + chunk; i++)
           a[i] = tid + 1;
   for (i = 0; i < n; ++i) cout << a[i] << " ";
00111222333
0 00011223300 
11122233300
```