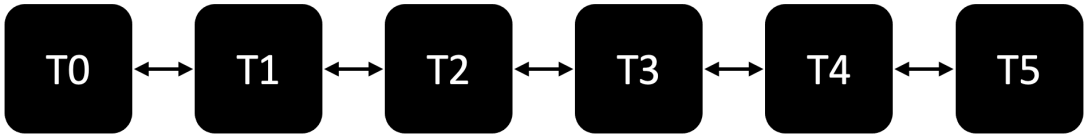
Test practic 2 APD

Testul practic este format din două cerințe care se punctează individual, cu mențiunea că a doua cerință este dependentă de rezolvarea primei cerințe.

Cerința 1 (6p). Să se implementeze, **folosind C/C++ și MPI** și presupunând **o topologie cu N task-uri conectate în linie** crescător după rang, algoritmul de sortare OETS pentru N elemente. Astfel, **fiecare task MPI va conține inițial câte un număr**. La finalul execuției programului vostru, valorile vor fi sortate **descrescător**: task-ul cu rangul 0 va avea numărul cel mai mare, task-ul cu rangul 1 va avea următorul număr, etc. După rularea algoritmului de sortare, fiecare task își va afișa rangul și valoarea sa finală (pe formatul *rang - valoare*).

Cerința 2 (4p). Odată ce numerele au fost sortate corect folosind OETS, se cere colectarea valorilor la **task-ul N - 1** și afișarea acestora în ordine **descrescătoare** (fără a se mai sorta o dată valorile venite). Comunicarea se poate realiza doar între task-uri vecine.

Puteți observa în imaginea de mai jos un exemplu de topologie pentru 6 task-uri, unde, de exemplu, task-ul 0 poate să comunice direct doar cu task-ul 1, task-ul 1 poate sa comunice direct doar cu task-urile 0 și 2, etc.



Algoritmul OETS este prezentat în laboratorul 3 și are următorul pseudocod pentru situația în care fiecare task are câte un număr:

```
for i = [0, n)
    if (i % 2 == 0)
        task-urile pare compara valoarea cu vecinul drept si pastreaza maximul
        task-urile impare compara valoarea cu vecinul stang si pastreaza minimul
    else
        task-urile impare compara valoarea cu vecinul drept si pastreaza maximul
        task-urile pare compara valoarea cu vecinul stang si pastreaza minimul
```

În urma compilării, va rezulta un executabil numit *testapd2*, care se va putea rula prin intermediul comenzii *mpirun -np N ./testapd2*. Fiecare task va avea valoarea inițială setată la **2R** (unde R este rangul task-ului curent). Aveți mai jos trei exemple de rulare, unde output-ul cu albastru este cel necesar pentru prima cerință, iar cel cu roșu este specific pentru a doua cerință. Afișarea trebuie realizată pe formatul prezentat.

```
$ mpirun -np 5 ./testapd2
3 - 2
1 - 6
0 - 8
4 - 0
2 - 4
8 6 4 2 0
```

```
$ mpirun -np 6 ./testapd2
0 - 10
1 - 8
3 - 4
2 - 6
5 - 0
4 - 2
10 8 6 4 2 0

$ mpirun -np 2 ./testapd2
1 - 0
0 - 2
2 0
```

Atenție! Pentru prima cerință, este suficient să afișați doar liniile cu albastru, din fiecare task (deci ordinea afișării lor pe ecran nu va fi neapărat aceeași ca în exemplele de mai sus). Pentru a doua cerință, doar task-ul N - 1 are voie să afișeze.

Atenție! Comunicația se va face strict cu task-urile vecine (cel de dinainte și cel de după task-ul curent, adică R - 1 și R + 1, dacă există). Dacă se realizează comunicație între task-uri neconectate în topologie la vreuna din cerințe, punctajul pe acea cerință va fi 0.

Atenție! O soluție care nu folosește ideea de la OETS (interschimbări între elemente de pe poziții cu paritate diferită) sau care nu respectă alte specificații ale enunțului (cum ar fi N task-uri pentru N elemente) NU va fi punctată.

Atenție! La rezolvarea cerinței 1, fiecare task va memora cel mult 2 valori la un moment dat.

Programul trebuie să funcționeze corect pentru orice valoare a lui N care respectă 2 <= N <= 12.

Trebuie să încărcați pe Moodle o arhivă care să conțină **în rădăcină** fișierele sursă, un Makefile cu target-ul *build* și eventual un fișier README.

Atenție! Fișierul README este necesar doar în cazul în care considerați că este nevoie de explicații adiționale în legătura cu implementarea realizată.