Corso di Ingegneria Elettronica
Insegnamento di "Strutture Software 1" a.a. 2008-09

Esercitazione n. 6

1. Definire la classe `SortedLinkedList` che estende la classe `LinkedList` (vista a lezione) e che mantiene ordinata la lista: ridefinire il metodo `add()` affinché inserisca gli elementi in ordine. Gli oggetti da inserire devono essere confrontabili e quindi devono implementare l'interfaccia `IComparable`. Si ordini la lista in modo che il valore massimo sia in testa (`first`) e il valore minimo in coda (`last`). Verificare la correttezza dell'implementazione: per esempio usando sequenze di stringhe in ordine, in ordine inverso, casuali, uguali. Verificare con sequenze di 21 elementi interi casuali, con valori compresi nell'intervallo [-30,30].

2. Sviluppare una classe `Process` che descriva un processo. Definire tre campi: un intero che identifica il processo (`pid`), una stringa per il nome del processo e un intero per l'occupazione di memoria. Definire il costruttore della classe e il metodo `ToString()`. Implementare l'interfaccia `IComparable`: il confronto è fatto sulla memoria utilizzata.

3. Scrivere una classe (che contiene il `Main()`) in cui creare un oggetto di tipo `SortedLinkedList`, lista di oggetti `Process`. La sequenza di processi "in ingresso al sistema" è contenuta nel file `processes.txt`: ogni riga contiene i dati di un processo, cioè il pid, il nome e la memoria occupata. Per simulare l'occupazione della memoria del sistema in ogni istante, eseguire le seguenti operazioni:

    - Leggere una riga del file ed inserire il processo nella lista.
    - Se la quantità di memoria utilizzata da tutti i processi presenti nella lista supera un valore di soglia (100000) eliminare dalla lista il processo che in quel momento occupa più memoria (utilizzare gli iteratori, oggetti `IEnumerator`).
    - Salvare su file il numero di processi presenti nel sistema e la quantità di memoria utilizzata da tutti i processi in quell'istante ("lettura di una riga").
    - Ripetere i precedenti passi per ogni riga del file.

Fare i grafici dell'occupazione di memoria e del numero di processi (in funzione degli istanti temporali). Commentare.
Valutare il costo computazionale per mantenere la lista ordinata e per eliminare il processo che occupa più memoria.