Algorytmy i struktury danych
LABORATORIUM

Zajęcia 8
Struktury danych: kolejka priorytetowa, kopiec.

Cel zajęć

Zapoznanie studentów ze strukturami danych kolejki priorytetowej oraz kopca, implementacja podstawowych operacji na tych strukturach danych, jak również implementacja wybranych problemów algorytmicznych wykorzystujących te struktury.

Zadania

1. Napisz funkcję, sprawdzającą, czy dana tablica liczb całkowitych jest kopcem: `boolean isHeap(int[] T)`. Dla danej tablicy T=[0, 12, 3, 7, 11, 10, 8, 2, 10, 9, 15] sprawdź, czy reprezentuje ona kopiec. Wykorzystaj funkcję `isHeap()`. Wydrukuj stosowny komunikat.
2. Przedstaw kopiec uzyskany po wstawieniu znaków: ALFAROMEO (w tej kolejności) do początkowo pustego kopca z łatwym dostępem do maksimum.
3. Wyznacz wszystkie kopce, które można utworzyć na podstawie pięciu znaków A B C D E.
4. Zbuduj klasę prywatną `KolejkaPriorytetowa`, wewnątrz której zaimplementuj taka kolejkę w postaci kopca, reprezentowanego jako tablica N elementów. Każdy element będzie obiektem typu `Pacjent`, reprezentowanym przez imię (typ `String`), nazwisko (typ `String`) i datę urodzenia (typ `Date`). Wewnątrz klasy zaimplementuj dwie funkcje: `void insert(Pacjent p)` – wstawienie do kolejki, Pacjent `delMax()` – pobranie i usunięcie elementu maksymalnego z kolejki (maksymalny jest tutaj rozumiany jako najstarszy). Następnie dodaj do kolejki 8 pacjentów:
   - Jan, Kowalski, 1961-09-20
   - Tamara, Bykowska, 1929-01-10
   - Marian, Baranowski, 1958-12-05
   - Katarzyna, Makowska, 1972-05-07
   - Joanna, Groth, 1942-07-15
   - Monika, Włodarska, 1964-02-27
   - Kazimierz, Nowakowski, 1937-03-21
   - Waldemar, Chamerski, 1978-11-11

   I wydrukuj powstałą kolejkę.
   Dokonaj symulacji obsługi trzech (najstarszych) pacjentów, a następnie dodaj kolejnego przybyłego (Anna, Maliszewska, 1981-08-03). Po każdej operacji pobrania/dodania elementu wydrukuj aktualną kolejkę. Jak rozwiążesz problem stałego rozmiaru tablicy?
5. Mając 8 pacjentów, jak w zad. 4, posortuj ich niemalejąco, od najmłodszych do najstarszych, wykorzystując sortowanie przez kopcowanie (*heapsort*).