FEJLETT PROGRAMOZÁSI NYELVEK, 2009
12. GYAKORLAT – Különleges tárolók

Cél:

- `bitset (#include <bitset>)`
- `stack (#include <stack>)`
- `priority_queue (#include <queue>)`

1. A `bitset<>` tároló

Készítsen programot, amely hosszú egészeket olvas a standard bemenetről és bitszám-összeg szerinti növekvő sorrendben kiírja a standard kimenetre.

Követelmények:

(a) Használjuk a **`bitset`** típust a bitszámösszeg számítására

(b) Az összehasonlítást végezzük **függvényobjektummal**

```
class BitSumCompare{
public:
    bool operator()( const bitset<32>& n1, const bitset<32>& n2) const{
       //Implementálni!
    }
};

multiset< bitset<32>, BitSumCompare > s;
```

Bemenet: `99 33 66 11 9 8 7 6`

Kimenet:

```
00001000 :8
00100001 :33
01000010 :66
00001001 :9
00000110 :6
00001011 :11
00000111 :7
01100011 :99
```

2. A `stack<>` tároló

Készítsen programot, amely fordított lengyel alakban megadott kifejezéseket értékel ki. A kifejezés csak a négy bináris műveletet tartalmazhatja: +, ,*, /, valamint valós operandusokat. Az egyszerűség kedvéért a bemenetben az entitások fehér karakterekkel vannak elválasztva.
Követelmény: Használjuk a `stack<T>` tárolót operandusveremként.

```
Bemenet: 1 2 + 4 2 - *
Kimenet: 6
```

FEJLETT PROGRAMOZÁSI NYELVEK, 2009
12. GYAKORLAT – Különleges tárolók

3. A `priority_queue<>` tároló

Készítsen programot, amely egy szövegállományban szereplő karaktereknek elkészíti a Huffman kódolását.
http://www.siggraph.org/education/materials/HyperGraph/video/mpeg/mpegfaq/huffman_tutorial.html

Követelmény:

Kötelező a `priority_queue` tárolót használni

Bemenet:

AAAAAAAAAAAAAA
BBB
C
D
E
F
GGG
HHHH

Példa helyes kimenetre:

Char	Freq.	Code
A	14	0
B	3	100
G	3	101
H	4	110
C	1	11100
F	1	11101
D	1	11110
E	1	11111