500. DD Megoldó Alfréd DD 500.

Programozás alapjai 2. (inf.) zárthelyi 2009.05.21. gyakorlat: /	Érdemjegy:
QBX734 0 **IB319/32.**	Hftest:

F.	Max.	Elért	
1	3		
2	4		
3	4		
4	4		
5	5		
Σ	**20**		

Minden beadandó megoldását a feladatlapra, a feladat után írja! *A megoldások során feltételezheti, hogy minden szükséges input adat az előírt formátumban rendelkezésre áll.*
A feladatok megoldásához csak a letölthető C és C++ összefoglaló használható. Számítógép, notebook, menedzser kalkulátor, organiser, rádiótelefon nem használható.

*A feladatokat **figyelmesen olvassa** el, megoldásukhoz **ne használjon fel STL** tárolót, kivéve, ha a feladat ezt külön engedi! **Ne írjon felesleges** függvényeket ill. kódot, a feladatra koncentráljon! Jó munkát!* *Értékelés: 0-8:1, 9-11:2, 12-14:3, 15-17:4, 18-20:5*

1. Feladat 3 pont

Mit ír ki a szabványos kimenetre az alábbi program? Válaszához használja a négyzetrácsos területet!

```
#include <iostream>
using namespace std;
class A {
  const char *s;
public:
  A(const char *s = "D") :s(s)   { cout << s << 'k'; f(); }
  virtual ~A()                   { cout << 'd'; }
  virtual void f()               { cout << "f";}
};
class B : public A {
public:
  B(const char *s ="C++")  { cout << 'K'; f(); }
  B(const B& b)            { cout << 'C'; f();}
  B(const A)               { cout << 'M'; }
  ~B()                     { cout << 'D'; }
  void f()                 { cout << "R";}
};
int main() {
  A me6("QBX734");   cout << endl;
  B b1("6");         cout << endl;
  B b2 = b1;         cout << endl;
  B b3(me6);         cout << endl;
  delete new A;      cout << endl;
  return(0);
}
```

Q	B	X	7	3	4	k	f							
D	k	f	K	R										
D	k	f	C	R										
D	k	f	M	d										
D	k	f	d											
D	d	D	d	D	d	d								

A 2. feladat megoldáshoz felhasználhatja az STL *deque* sablonját, melynek megadjuk a feladat megoldása szempontjából fontosabb tagfüggvényeit:

(constructor)	Construct deque container
oparetor=	Copy container content
size	Return size
empty	Test whether container is empty
front	Access first element
back	Access last element

push_back	Add element at the end
push_front	Insert element at beginning
pop_back	Delete last element **(void!)**
pop_front	Delete first element **(void!)**
insert	Insert elements
erase	Erase elements

2. Feladat Σ 4 pont

Egy Queue*6* osztályt kell létrehoznia, ami karakterek dinamikus tárolására alkalmas. Az osztály publikus interfészét az alábbi kommentezett deklaráció rögzíti. A deklaráción már **csak a pontozott részen lehet változtatni,** ill. a megjelölt tagfüggvényeket áthúzhatja! Követelmény, hogy az osztály legyen átadható érték szerint függvényparaméterként, és működjön helyesen a többszörös értékadás is.

```
class Queue6 {
public:
  Queue6(const char *p = 0);// Nullával lezárt stringre mutató pointer alapján létrehozza a sort
  Queue6(const Queue6 &); // Másoló konstr. ..... ÁTHÚZHATJA, ha nem kell.
  Queue6 &operator=(const Queue6 &); // Értékadó operátor .....ÁTHÚZHATJA, ha nem kell.
  void push_back(char c);// A sor végére tesz egy karaktert
  char remove_back();// Kiveszi a legutolsó karaktert és visszadja. Ha a sor üres, underflow_error() kivetelt dob.
  void push_front(char c);// A sor elejére tesz egy karaktert
  char remove_front();// Kiveszi a legelső karaktert és visszadja. Ha a sor üres, underflow_error() kivetelt dob.
  bool empty() const;    // true, ha üres a sor
  int size()const;       // Tárolt karakterek számat adja
  ~Queue6 ();
private:
   deque<char> q;
    ........................
};
```

Egészítse ki a fenti deklarációt! (1p)

A tagfüggvények közül **valósítsa meg** a következőket:..... (3 pont)!

A megoldáshoz felhasználhatja az STL *deque* sablonját (ld. első oldal alján).

A feladat lényegében a 11-es sorszámmal lekérdezhető *hftest* feladattal azonos. Az ahhoz hasonló megoldás jár a legkevesebb kódolással:

1. Fel kell venni egy deque<char> adattagot. Ez a tároló mindent tud, ami kell a faladathoz, csak nem pontosan úgy. Az alapértelmezett másoló operátor és az operator= pontosan jó, ezért ki kell húzni, ha nem húznánk ki, meg kellene valósítani.
2. Ezután meg kell valósítani a feladatban szereplő tagfüggvényeket:

konstruktor:

```
Queue6::Queue6(const char *p) {
    if (p != NULL) {                // figyel a null-ra 0.5 pont
          const char *pe = p;
          while (*pe) pe++;
          q = deque<char>(p, pe);   // felépiti a strringet valahogy 0.5p
    }
}
```

remove_back, push_front:

```
char Queue6::remove_back() {
    if (q.empty()) throw range_error("Ures queue");
    char c = q.back();
    q.pop_back();       // tdudja (odairtuk), hogy az STL-van void a pop 0.5p
    return c;           // visszadja a karaktert 0.5
}
void Queue6::push_front(char c) {
    q.push_front(c);    // elteszi a karaktert 0.5p
}
```

remove_front, push_back:

```
char Queue6::remove_front() {
    if (q.empty()) throw range_error("Ures queue");
    char c = q.front();
    q.pop_front();      // tdudja (odairtuk), hogy az STL-van void a pop 0.5p
    return c;           // visszadja a karaktert 0.5p
}
void Queue6::push_back(char c) {
    q.push_back(c);     // elteszi a karaktert 0.5p
```

```
}
```

3. Feladat Σ 4 pont

Tételezze fel, hogy **a 2. feladat *Queue6*** osztálya elkészült és hibátlanul működik! Ezen osztály felhasználásával készítsen egy olyan ***UdvariasQueue*** osztályt, ami kompatibilis mind a **Queue*6*** osztállyal, mind a következő (már nem módosítható) osztállyal (1p):

```
class Bemutatkozo {     // ez az osztály nem módosítható!
      string uzenet;
public:
      Bemutatkozo (string const & s) : uzenet(s) {}
      void bemutatkozik(ostream &o) const { o << uzenet; }
};
```

Legyen az UdvariasQueue osztálynak két további függvénye: **`print`**, amely kiírja egy adott ostream-re az üzenetet és a Queue méretét (1p), és **`toString`**, amely az előbbi üzenetet egy string típusú objektumban adja vissza, és nem ír ki semmit. (1p) Feltételezheti, hogy a string osztály minden char * típushoz kötődő funkciót, és a + += = operátorokat értelemszerűen megvalósítja. A megoldást nem értékeljük, amennyiben nem használja fel a *Queue* és *Bemutatkozo* osztályokat!

Írjon egy programrészletet, ami beolvas 10 karaktert a szabványos bementről, és eltárolja egy *UdvariasQueue* –ban, majd kiírja fordított sorrendben a szabványos kimenetre. A kiírás előtt a queue „mutatkozzon be”! (1p)

Egy lehetséges megoldás:

```
class UdvariasQueue : public Bemutatkozo, public Queue6 {
public:
    UdvariasQueue(const char *uzen = "", const char *p = 0)
            :Bemutatkozo(uzen), Queue6(p) {}
    void print(ostream&);
    string toString();
};

void UdvariasQueue::print(ostream& os) {
      bemutatkozik(os);
      os << " " << "Meretem:" << size();
}

string UdvariasQueue::toString() {
      ostringstream ss;
      print(ss);
      return ss.str();
}

int main() {
      UdvariasQueue udvarias("Hello queue vagyok!");
      udvarias.print(cout);
      cout << endl;
      for (int i = 0; i < 10; i++) {
        char c;
        cin >> c;
        udvarias.push_front(c);
      }
      while (!udvarias.empty())
        cout << udvarias.remove_front();
      return 0;
}
```

4. Feladat Σ 4 pont

Készítsen függvénysablont (**my_positive**), ami eldönti a paraméterként kapott generikus adatról, hogy az nagyobb-e mint 0! Tudjuk, hogy a generikus adatra működik a **`>(double)`** operátor, vagyis létezik a

`bool T::operator>(double)` függvény! (1 pont)

Adott a következő deklaráció: **`Tarolo tar;`** Tételezze fel, hogy a **Tarolo** osztály long számokat tárol, és van iterátora, valamint létezik a szokásos begin() ill. end() tagfüggvénye is! Írjon olyan programrészletet, amely a **my_positive** sablon felhasználásával kiírja a **`tar`** objektumban tárolt számok közül a nullánál nagyobb értékeket! (1 pont)

Készítsen generikus függvényt (**my_count_if**), amely az STL *count_if* algoritmusához hasonlóan megszámolja a paraméterként kapott iterátorok között adott predikátumfüggvénynek eleget tevő elemeket! Működjön helyesen a következő programrészlet, felhasználva az eddigi sablonokat: (2 pont)

```
int v[] = { 1, 2, 3, -77, 88 };
cout << my_count_if(v, v+5, my_positive<int>);
```

Egy lehetséges megoldás:

```
template<class T>
bool my_positive(T a) {
      return a > 0;
}
```

```
for (Tarolo::iterator i1 = tar.begin(); i1 != tar.end(); ++i1)
      if (my_positive(*i1))
            cout << *i1;
```

```
template<class T, class S>
int my_count_if(T from, T to, S sel) {
      int count = 0;
      for (; from != to; ++from)
            if (sel(*from)) count++;
      return count;
}
```

5. Feladat Σ 5 pont

C++ nyelven szeretnénk modellt készíteni a következő leírás alapján. Egy óvodás **hátizsák**jába sok (max. 100) **játék** fér. Minden játéknak van népszerűségi indexe (egész szám, hány óvodás irigykedik a játék tulajdonosára), és minden játékot meg lehet nyomni (*push()*), amire minden játék máshogy reagál. Játékból több fajta is van, és a játékfajták száma később csak nőni fog. Játékok jelenleg az alábbiak:

- Macintosh-**MACi** (index: 15, verziószám: egész). Ha megnyomják, és a verziószám legalább 10, akkor kiírja, hogy „Én is unix vagyok”.
- Linux-**pingvin** (kezdő index: 500, kernel_verzió: sztring), ha megnyomják, kiírja a kernel verziószámát, és ha az index kisebb, mint 97, akkor 4-gyel megnöveli. Ha megsemmisül, kiírja, hogy „UNIX4EVER”.
- Microsoft-**majom** (kezdő index: 53, vagyon: long double). ha megnyomják, kiírja, hogy „I! LOVE! THIS! COMPANY!!!”.

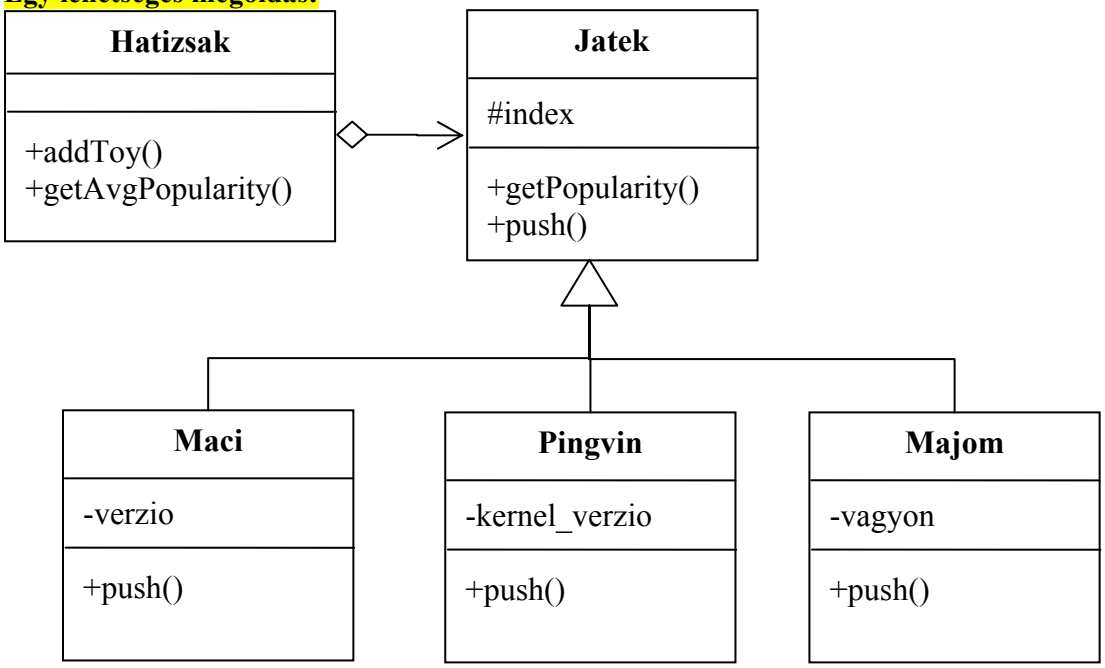
A rendszerben a következő funkciókat kell megvalósítani:

- hátizsák átlagos népszerűségi indexének lekérése (*getAvgPopularity()*)
- hátizsákba új játék behelyezése (*addToy()*)
- játék népszerűségének lekérdezése (*getPopularity()*)
- minden játékhoz a megnyomás (*push()*)
- pingvin destruktora.

Feladatok:

- **Tervezzen** meg egy olyan OO modellt, mely a fenti követelményeket kielégíti! Rajzolja fel a modell osztálydiagramját! (0,5p)
- **Deklarálja** a *Hátizsák, Játék, Maci, Pingvin* osztályokat és az elvárt funkciók ellátásához szükséges tagfüggvényeket! Használja a dőlt betűs neveket! (2)
- **Implementálja** a fenti osztályokat és azok tagfüggvényeit! Az osztályokat olyan módon készítse el, hogy újabb játék-típus felvételekor a már meglevő kódot ne kelljen módosítani! (1,5 pont)
- **Írjon** egy egyszerű programrészletet, ami készít egy hátizsákot, és elhelyez benne egy pingvint (kernel verzió: 2.6.11) és egy majmot (vagyona $3.5e274). Írassa ki a hátizsák átlagos népszerűségét, majd nyomja meg a pingvint, és így is írja ki a hátizsák népszerűségét! (1 pont)

Egy lehetséges megoldás:




```
class Jatek {
protected:
  int index;
public:
  Jatek(int i) : index(i) {}
  int getPopularity() {
    return index;
  }
  virtual void push() = 0;
  virtual ~Jatek() {}
};
class Maci : public Jatek {
  int verzio;
public:
  Maci(int v) : Jatek(14), verzio(v) {}
  void push() {
    if (verzio >= 10)
      cout << "En is unix vagyok!"
           << endl;
  }
};
class Pingvin : public Jatek {
  string ker_ver;
public:
  Pingvin(string k) : Jatek(43),
                      ker_ver(k) {}
  void push() {
    cout << ker_ver << endl;
    if (index < 97) index += 4;
  }
  ~Pingvin() {
    cout << "UNIX4EVER" << endl;
  }
};
```

```
class Hatizsak {
  Jatek* tar[100];
  int n;
public:
  Hatizsak() : n(0) {}
  ~Hatizsak() {
    for (int i = 0; i < n; i++) {
      delete tar[i];// ha adoptal
    }
  }
  void addToy(Jatek* j) {
    if (n < 100) {
      tar[n] = j;
      n++;
    } else {
      delete j; // ha adoptal
    }
  }
  int getAvgPopularity() {
    int sum = 0;
    for (int i = 0; i < n; i++) {
      sum +=
        tar[i]->getPopularity();
    }
    return sum/n;
  }
};
// a feladat szerint nem kell
class Majom : public Jatek {
  long double vagyon;
public:
  Majom(long double ld) :
       Jatek(43), vagyon(ld) {}
  void push() {
    cout << "I! LOVE! THIS!\
COMPANY!!!"
         << endl;
  }
};
```

```
int main() {
  Hatizsak h;
  Pingvin* p = new Pingvin("2.6.11");
  Majom* m = new Majom(3.5e274);
  h.addToy(p);
  h.addToy(m);
  cout << h.getAvgPopularity() << endl;
  p->push();
  cout << h.getAvgPopularity() << endl;
  return 0;
}
```