```cpp
#include <iostream>
#include <cstring>

using std::cout;
using std::endl;

class Allat {
private:
  char *nev;
protected:
  enum { tele, ures } bendo;
public:
  Allat(const char *knev) : bendo(ures) {
    nev = new char[strlen(knev) + 1];
    strcpy(nev, knev);
  }
  bool isEhes() { return bendo==ures; }
  const char *getNev() { return nev; }
  virtual void jatszik(Allat *pajtas) = 0;
  virtual void hallat() = 0;
  virtual ~Allat() {
    cout<< "~" << nev << endl;
    delete [] nev;
  }
};

class Majom : public Allat {
protected:
  static const char* HANG;
public:
  Majom(const char *knev) : Allat(knev) { }
  void jatszik(Allat *pajtas) {
    this->hallat();
    pajtas->hallat();
  }
  void hallat() { cout << HANG << ":" << getNev() << endl; }
  ~Majom() { cout << HANG << HANG << " "; }
};

const char* Majom::HANG = "Mak";

class Tigris : public Allat {
public:
  Tigris(const char *knev) : Allat(knev) { }
  void jatszik(Allat *pajtas) {
    delete pajtas;
    bendo = tele;
  }
  void hallat() { cout << "HRRR" << endl;}
};


```

```
53  class CirkusziTigris : private Tigris {
54  public:
55    CirkusziTigris(const char *knev) : Tigris(knev) { }
56    void hallat() {
57      Tigris::hallat();
58      cout << getNev() << " vagyok, idomított tigris.\n";
59    }
60  };
61
62  int main() {
63    Allat *cicus = new Tigris("Cicus");
64    Allat *csita = new Majom("Csita");
65    Allat *tarzan = new Majom("ÁIÁIÁIÁ");
66    cout << "-= Csita játszik =-" << endl;
67    csita->jatszik(tarzan);
68    csita->jatszik(cicus);
69    cout << "-= Cicus játszik =-" << endl;
70    cicus->jatszik(csita);
71    cicus->jatszik(tarzan);
72    cout << "-= Attrakció =-" << endl;
73    CirkusziTigris *ubul = new CirkusziTigris("Ubul");
74    ubul->hallat();
75    delete cicus;
76    delete ubul;
77  }
```

---

```
#include <iostream>
using namespace std;
class A {
protected:
    int k;
public:
    A(const int i = 0) :k(i){ cout << 'k'; }
    A(const A& a) { k = a.k; cout << 'c'; }
    void operator=(A& a) { k = a.k; f(a); cout << 'e'; }
    A& operator*(int i) { cout << i*100; return *this; }
    virtual void f(A a) { k++; cout << 'f'; }
    ~A() { cout << 'd'; }
};
class B :public A {
    int k;
public:
    B(const int i = 0) :k(i){ cout << 'K'; }
    B(B& b) :k(b.k) { cout << k << 'C'; }
    B& operator*(int i) { cout << i; return *this; }
    void f(A a) { A::f(a); cout << 'F' << A::k; }
    ~B() { cout << 'D' << k; }
};
B& operator*( int i, B& b ) { cout << i; return b; }
void main() {
    B b(12); cout << '\n';   // ..............
    A a = b; cout << '\n';   // ..............
    b = b * 2; cout << '\n'; // ..............
    b = 3 * b; cout << '\n'; // ..............
}                            // ..............
```