Lab_10

Dla podanych klas A, B, D

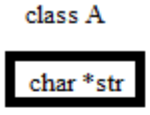
```
// Lab_10.cpp : This file contains the 'main' function. Program execution begins and ends there.
//

#include <iostream>

using namespace std;

static int count_alloc = 0;  // counter - how many times memory was allocated

class A
{
	char* str;
public:
	A(char* sstr);
	~A();
	friend ostream& operator << (ostream& strm, const A& ob);
};

A::A(char* sstr)
{
	try {
		str = new char[128];
		count_alloc++;
		memset(str, 0, _msize(str));
		strcpy_s(str, _msize(str), sstr);
	}
	catch (bad_alloc) {
		cerr << errallocmem;
	}
}

A::~A()
```

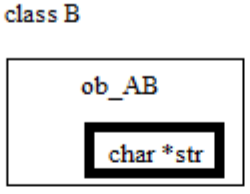
```
{
	cout << "~A(): " << str << "\n";
	if (str)
		delete[] str;
	count_alloc--;
	str = NULL;
}

ostream& operator << (ostream& strm, const A& ob)
{
	strm << "   " << ob.str << endl;
	return strm;
}

class B
{
protected:
	A ob_AB;
public:
	B(char* sstr) : ob_AB(sstr) {}
	~B();
	virtual void disp() { cout << typeid(*this).name() << ob_AB; }
};

B::~B()
{
	cout << "~B()\n";
}


class D : public B
{
protected:
	A ob_AD;
public:
	D(char* sstr1, char* sstr2) : B(sstr1), ob_AD(sstr2) {};
	~D();
	virtual void disp() override { cout << typeid(*this).name() << ob_AB << ob_AD << endl; }
```

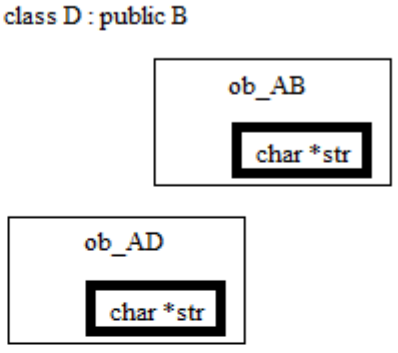
```
};


D::~D()
{
	cout << "~D()\n";
}


int main()
{
	B* ptr_BB = NULL;
	try {
		B* ptr_B = new B("ob_AB");
		D* ptr_D = new D("ob_AB", "ob_AD"); //The object of derived class arises
		count_alloc += 2;
		ptr_BB = ptr_B;                  //pointer of the base class type points to an object of the base class type
		ptr_BB->disp();
		ptr_BB = ptr_D;                  //pointer of the base class type points to an object of the derived class type
		ptr_BB->disp();

		//delete ptr_D;                  //No dynamic polimorphism
		//ptr_D = NULL;

		cout << "ptr_BB points to D: delete ptr_BB\n";
		delete ptr_BB;                   //what destructors will be called?
		ptr_BB = ptr_B;                  //pointer of the base class type points to an object of the base class type

		cout << "\nptr_BB points to B: delete ptr_BB\n";
		delete ptr_BB;                   //what destructors will be called?
		count_alloc -= 2;

		if (count_alloc)
			cout << "error: leak of memory !!!!!!!!!!!!!!!!!!! \n";
	}
	catch (bad_alloc) {
		cerr << errallocmem;
	}
```

```
        system("pause");
        return 0;
}
```

Ten program traci pamięć. Znajdź błąd i popraw go. Napisz kod dla errallocmem użytkownika-manipulatora .







Tworzenie obiektu D

1. konstruktor B
2. konstruktor A - obiekt ob_AB
3. konstruktor D
4. konstruktor A - obiekt ob_AD

Niszczenie obiektu D

1. Destruktor D
2. destruktor A - obiekt ob_AD
3. destruktor B
4. destruktor A - obiekt ob_AB