Lab_10

For given classes A, B, D

```
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() { cout << typeid(*this).name() << ob_AD << endl; }
};


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

Follow the next steps in the main() function:

```
int main()
{
	B* ptr_BB = NULL;
	try {
```

```
            B* ptr_B = new B("B");
            D* ptr_D = new D("B", "D"); //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;                           //wpointer of the base class type points to an object of the derived class type
            ptr_BB->disp();
            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

            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;
}
```

This program is losing memory. Find the bug and correct it. Write code for user-manipulator `errallocmem`.