1. 2 נק' מחלקת הבסיס מכילה מתודה pure virtual שלא מימשנו במחלקה הנגזרת. האם ניתן ליצר אובייקטים מסוג המחלקה הנגזרת? יש לנמק.

לא ניתן ליצור אובייקט מסוג המחלקה הנגזרת מכיוון שלא נממשנו את מתודה pure virtual אז גם המחלקה הנגזרת הופכת להיות ABSTRACT ולכן לא ניתן ליצור אובייקט! לא זו אבא ולא זו בן.

2. 2 נק' הסבירו מה ההבדל בין קישור סטטי לקישור דינאמי ותארו מתי קורה כל אחד מהם.

קישור סטטי מתבצע בזמן קומפילציה וקישור דינאמי קורה בזמן ריצה

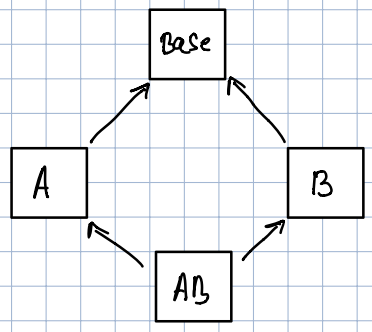
קישור סטטי – משתמש בו כאשר אני בוחרים שב תת פונ שלנו ירצה להשתמש מהמחלקה שלנו ומתודה ולא ירצה ישתמש (overloading)

קישור דינאמי – יתכן כי אחרי מבאנאי המחלקה יהיה מישהו אחר שדרס לבחור בזמן ריצה!

```
// C++ program to illustrate the concept of dynamic binding
#include <iostream>
using namespace std;

class B
{
    public:

    // Virtual function
    virtual void f()
    {
        cout << "Base class function called.\n";
    }
};

class D: public B
{
    public:
    void f()
    {
        cout << "Derived class function called.\n";
    }
};

int main()
{
    B base;
    D derived;

    B *basePtr = &base;
    basePtr->f();

    basePtr = &derived;
    basePtr->f();

    return 0;
}
```

```
// C++ program to illustrate the concept of static binding
#include <iostream>
using namespace std;

class ComputeSum
{
    public:

    int sum(int x, int y)
    {
        return x + y;
    }

    int sum(int x, int y, int z)
    {
        return x + y + z;
    }
};

int main()
{
    ComputeSum obj;
    cout << "Sum is " << obj.sum(10, 20) << '\n';
    cout << "Sum is " << obj.sum(10, 20, 30) << '\n';

    return 0;
}
```

3. 2 נק' נניח שנתונה מחלקה base שממנה יורשים באופן וירטואלי מחלקות A ו- B. נניח כי נתונה מחלקה AB שיורשת מA ו- מ- B. מי מאתחל את הבנאי של base?

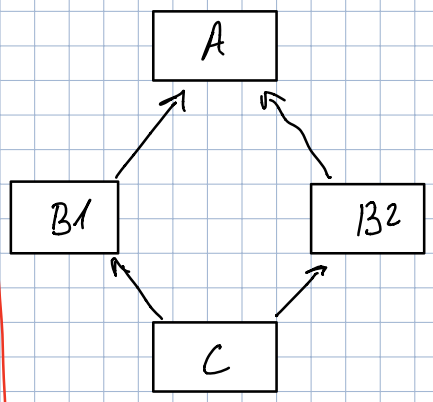


האובייקט ברמה התחתונה (AB) אומר לקרוא לבנאי של המלא הקדמון (Base) שמגדיר מתבצעת הירשה וירטואלית למי שהוא קורא לבנאי של אבא שלו

4. 6 נק' נתון הקוד הבא:

```
#include <iostream>
using namespace std;

class A{
   public:
  A() {cout<<"A::C'tor"<<endl;}
  virtual ~A() {cout<<"A::D'tor"<<endl;}
  virtual void func1() {cout<<"A::func1()"<<endl;}
  virtual void func2(){cout<<"A::func2()"<<endl; func1();}
  virtual void func3() = 0;
};

class B1 :virtual public A{
    public:
    B1() {cout<<"B1::C'tor"<<endl;}
    virtual ~B1() {cout<<"B1:D'tor"<<endl;}
    void func1() {cout<<"B1::func1()"<<endl;}
    virtual void func3() {cout<<"B1::func3()"<<endl; func2();}
};

class B2: virtual public A {
    public:
     B2() {cout<<"B2::C'tor"<<endl;}

    ~B2() {cout<<"B2::D'tor"<<endl;}
    void func2() const {cout<<"B2::func2()"<<endl;}
};

class C :public B1, public B2{
   public:
   C () {cout<<"C::C'tor"<<endl;}
   ~C() {cout<<"C::D'tor"<<endl;}
   virtual void func1() {cout<<"C::func1()"<<endl;}
   void func3() {cout<<"C::func3()"<<endl;}
};
```

לא אמיר מתיגר!



A class VTABLE	B1 class VTABLE	B2 class VTABLE	C class VTABLE
A::~A()	B1::~B1()	B2::~B2()	C::~C()
A::func1()	B1::func1()	A::func1()	C::func1()
A::func2()	A::func2()	A::func2()	A::func2()
A::func3()	B1::func3()	A::func3()	B1::func3()

```
int main()
{
  A* a = new B1();
  a->func2();
  B2* b2 = new C();
  b2->func3();
  delete a;
  delete b2;
  return 0;
}
```

הקצאה ?

A* a = new B1(); = A::ctor()
B1::ctor

a->func2 = A::func2();
B1::func1();

B2* b2 = new C(); = B2::ctor
C::ctor

b2->func3(); = C::func3()

delete a; = B1::D'stor
A::D'stor

delete b2: C::D'stor
B2::D'stor