Inheritance Experiments

Problem 1: Inheritance Trace

Inspect the following nonsense code where all implementations have been **`inline`**d for convenience (assume all methods are **`public`** and all instance methods are **`const`**):

```
class orion {
   virtual void jupiter() = 0;
   void mars() { cout << "orion::mars" << endl; neptune(); }
   virtual void neptune() { cout << "orion::neptune" << endl; uranus(); }
   virtual void saturn() = 0;
   static void uranus() { cout << "orion::uranus" << endl; }
};

class signus : public orion {
   virtual void jupiter() { cout << "signus::jupiter" << endl;}
   virtual void neptune() { cout << "signus::neptune" << endl; saturn(); }
   static void uranus() { cout << "signus::uranus" << endl; }
};

class vulpecula : public orion {
   virtual void jupiter() { cout << "vulpecula::jupiter" << endl; }
   virtual void neptune() { cout << "vulpecula::neptune" << endl;
                            orion::neptune(); }
   virtual void saturn() { cout << "vulpecula::saturn" << endl; }
   static void uranus() { cout << "vulpecula::uranus" << endl; }
};

class gemini : public signus {
   void mars() { cout << "gemini::mars" << endl; uranus(); }
   virtual void saturn() { cout << "gemini::saturn" << endl; mars(); }
};
```

The `explore` function is designed to take a **`const orion`** pointer as its only parameter:

```
static void explore(const orion *constellation)
{
   constellation->uranus();
   constellation->mars();
}
```

What are the possible types that **`constellation`** may be addressing at runtime? For each possibility, trace through a call to **`explore`** and present its output. Please use the next page for your answers.

Problem 2: Static versus Dynamic Type

You are given the following two classes and the **`test`** function:

```
class Vegetable {

   public:
      XXXX void grow();
};

class Squash: public Vegetable {

   public:
      XXXX void grow();
};

void test(YYYY veggie)
{
   veggie.grow();
}
```

The sequence **`XXXX`** can be replaced with either **`virtual`** or blank space and **`YYYY`** can be **`Vegetable`**, **`Vegetable*`**, or **`Vegetable&`** (assume the code within function is altered to match types when necessary, i.e. change **`.`** to **`->`**).

The table below enumerates the possible permutations between the two replacements. Assuming an object of type **`Squash`** (or its address for the pointer version) is passed to the **`test`** function, indicate for each entry in the table whether such a call compiles and if so, which version of the **`grow`** method is invoked, either **`Vegetable`** or **`Squash`**.

YYYY = / XXXX =	`Vegetable`	`Vegetable&`	`Vegetable*`
`virtual`			
`blank`			