1、下面代码的输出结果是什么？(D)

```
function sayHi() {
   console.log(name);
   console.log(age);
   var name = 'Lydia';
   let age = 21;
}
```

A. undefined 和 undefined;

B. Lydia 和 ReferenceError;

C. ReferenceError 和 21;

D. undefined 和 ReferenceError;

解析：本题的考点主要是 var 与 let 的区别以及 var 的预解析问题。var 所声明的变量会被预解析，var name;提升到作用域最顶部，所以在开始的 console.log(name)时，name 已经存在，但是由于没有赋值，所以是 undefined；而 let 会有暂时性死区，也就是在 let 声明变量之前，你都无法使用这个变量，会抛出一个错误，故选 D。

2、下面关于类 class 的描述，错误的是：（D）

A、 JavaScript 的类 class 本质上是基于原型 prototype 的实现方式做了进一步的封装

B、 constructor 构造方法是必须的

C、 如果类的 constructor 构造方法有多个，后者会覆盖前者

D、 类的静态方法可以通过类名调用，不需要实例化

3、请问以下代码 fn.name 的结果是什么？ （ C）

```
function Fn(){
      this.name = 'miaov';
      this.address = '北京';
      return {}
   }
   var fn = new Fn();
```

A. miaov

B. 北京

C. undefined

D. 北京 miaov

解析：new 命令的作用，就是执行构造函数，返回一个实例对象。当实例化对象的时候，构造函数默认隐藏返回一个 this 对象，this 指向新生成的实例对象，并且会执行构造函数中的代码。但是修改构造函数的返回值后，最后会返回修改的返回值.题中最终返回值是空对象，所以 fn.name 的结果是 undefined

4、关于 ES6 的使用以下描述错误的是？（B ）

A、

```
const a = 1;
```

```
const b = 2;

const map = {a, b};
```
B、
```
enum TYPE {
  OK,
  YES
}
```
C、
```
class A {
constructor (a) {
this.a = a;
}
}
class AA extends A {
constructor (a, b) {
super(a);
this.b = b;
}
toString () {
return this.a + '' + this.b;
}
}
```
D、
```
function* greet(){

yield
"How";

yield
"are";

yield
"you";

}

var greeter = greet();

console.log(greeter.next().value);
```

console.log(greeter.next().value);

console.log(greeter.next().value);
解析：

es6 中将构造方法的 function 换成了 class，

用于与普通函数区分，其中的属性都放在 constructor 中，

方法在原型中，子类继承采用 extends 关键字;对于 es6 中枚举的使用，

只能是以类的方式定义枚举类，不能直接使用 enum 关键字

5、下面运行结果正确的是（D ）

var a = {}, b = Object.prototype;
[a.prototype === b, Object.getPrototypeOf(a) === b]
A、other

B、[true, true]

C、[false, false]

D、[false, true]

解析：

这题考的是__proto__和 prototype 的区别

Object 实际上是一个构造函数（typeof Object 的结果为"function"）

使用字面量创建对象和 new Object 创建对象是一样的，

所以 a.__proto__也就是 Object.prototype，

而 Object.getPrototypeOf(a)与 a.__proto__是一样的，所以第二个结果为 true

而实例对象是没有 prototype 属性的，只有函数才有

所以 a.prototype 其实是 undefined，第一个结果为 false