1. 输出自己的学号

```
// const char*
// 不是sysy标准中定义的类型，sysy也没有extern这个声明外部函数的关键字，
// 另外sysy也没有... 这个不定长参数的语法
// 所以这个测试例基于一个sysy拓展的实现链接到glibc或者是其他c库实现打印
extern int printf(const char *fmt, ...);

void print(int num) { printf("%d", num); }

int main() {
  int num = 21051244;
  print(num);
  return 0;
}
```

sysy标准中没有定义字符串，sysy也没有extern这个声明外部函数的关键字，另外sysy也没有... 这个不定长参数的语法　所以这个测试例基于一个sysy拓展的实现最终链接到glibc或者是其他c库实现打印

2. 变量声明注释和隐式转换

```
// 注释
/*

多行注释

*/

int main() {
  // 变量声明
  int a = 1;
  float b = 2.0;
  // 允许隐式转换
  float c = a;
  int d = b;
  return a + b;
}
```

3. 常量声明， 全局变量声明， if分支语句

```
// 定义常量
const int I_DONT_KNOW = -1;
// 定义函数
int fib(int val) {
  // 分支语句
  if (val == 0) {
    return 0;
  } else if (val == 1) {
    return 1;
  } else {
    if (val == 2) {
      return 1;
```

```
    } else {
      if (val == 3) {
        return 2;
      } else {
        return I_DONT_KNOW;
      }
    }
  }
}
```

4. 数组声明， 函数声明， 数组索引

```
int a[9] = {};
int b[10][2] = {{}, {22}, {1, 2}};
// 数组构造
float c[10][2][3] = {{{1.0}}};

int foo() { return a[8] + b[0][0]; }
```

5. 作用域

```
int a = 3;
// sysy中并没有定义assert的实现, 这里的assert只是说明不同作用域中下的`a`是不同的

int main() {
  // 作用域
  int a = 2;
  {
    int a = 1;
    {
      int a = 0;
      assert(a == 0);
    }
    assert(a == 1);
  }
  assert(a == 2);
  return 0;
}
```

6.

```
int main() {
  int x = 2;
  int y = 1;
  int z = 3;
  int t;
  if (x > y) {
    t = x;
    x = y;
    y = t;
  }
  if (x > z) {
    t = z;
```

```
    z = x;
    x = t;
  }
  if (y > z) {
    t = y;
    y = z;
    z = t;
  }
  return 0;
}
```

7. 循环控制

```
extern int printf(const char *fmt, ...);
void print(int a, int b, int c) { printf("%d %d %d\n", a, b, c); }

int main() {
  int i = 0;
  int j = 0;
  // sysy标准中没有定义for 控制流语句,
  // for (i = 1; i <= 9; i++) {
  //   for (j = 1; j <= i; j++) {
  //     print(i, j, i * j);
  //   }
  // }
  while (i < 9) {
    i = i + 1;
    j = 0;
    while (j < i) {
      j = j + 1;
      print(i, j, i * j);
    }
  }
}
```