C语言实验四：函数和控制结构

必做题要求

1. **课程内完成**：OJ系统的F~N题
2. **提交要求**：提交如图所示的截图，**截图右上角必须包含你的学号信息**

Y	F	C语言实习题二——1.成绩判断	437	790
Y	G	C语言实习题二——2.字符转换	429	1004
Y	H	C语言实习题二——3.利用海伦公式求三角形面积	416	1767
Y	I	C语言实习题二——4.判断是否能构成一个三角形	405	592
Y	J	C语言实习题二——5.按从大到小排序三个数	404	1875
Y	K	C语言实习题三——1.数据统计	374	948
Y	L	C语言实习题三——2.爱因斯坦阶梯问题	374	635
Y	M	C语言实习题三——3.猴子吃桃问题	357	669
Y	N	C语言实习题三——4.求两个数的最大公约数和最小公倍数	329	939

选做题要求

- **题目**：任意进制转换器（详细要求见下文）
- **完成奖励**：
  - **晚上9点前完成**：可找我检查并提前离开
  - **晚上9:15前完成**：平时成绩总分加0.5分（加满为止）

📝 选做题：任意进制转换器

🎯 任务目标

请补全以下 C 程序，实现 **将一个给定进制的正整数字符串转换为另一个进制的字符串表示。**程序需支持 **2 到 16 进制** 之间的相互转换，其中数字 10~15 用大写字母 `'A'` 到 `'F'` 表示。
你将实现多个辅助函数，并最终通过 `Base2Base` 完成整个转换流程。

🔧 已提供函数（可直接使用）

为降低难度，以下两个函数已给出，供你参考和调用：

```c
// 将字符 '0'-'9','A'-'F' 转换为对应的整数值（0~15）
// 若字符非法（如 'G', 'z'），返回 -1
int Char2Int(char ch)
{
      if (ch >= '0' && ch <= '9')
            return ch - '0';
      switch (ch)
     {
            case 'A': return 10;
            case 'B': return 11;
            case 'C': return 12;
            case 'D': return 13;
            case 'E': return 14;
            case 'F': return 15;
            default:   return -1;
     }
}

```

```c
// 计算字符串长度（不使用 string.h）
int Strlen(char *str)
{
      int i = 0;
      while (str[i] != '\0')
            i++;
      return i;
}
```

> 💡 注意：输入字符串中**只包含合法的大写十六进制字符**（即 `'0'-'9'` 和 `'A'-'F'`），但你的程序仍应能处理如 `"0"`、单字符等边界情况。

📦 需要你实现的函数

请根据注释完成以下函数的定义：

```c
char Int2Char(int d);               // 将 0~15 转为 '0'-'9' 或 'A'-'F'
int Power(int base, int exp);   // 计算 base^exp（exp ≥ 0）
int Base2Dec(int base, char *num);          // 将 base 进制字符串转为十进制整数
void Dec2Base(int num, int base, char *converted); // 十进制转 base 进制（低位在前）
void Reverse(char *str);          // 反转字符串（原地操作）
void Base2Base(int base_from, int base_to, char *num_from, char *num_to);
```

要求Reverse函数除数组长度变量外不得开辟新的内存。

主函数逻辑（已给出，无需修改）：

```c
int main(void)
{
      int base_from, base_to;
      char num_from[11];
      char num_to[11];

      scanf("%d %s %d", &base_from, num_from, &base_to);
      Base2Base(base_from, base_to, num_from, num_to);
      printf("%s\n", num_to);
      return 0;
}
```

> ⚠️ 假设输入数据合法：`2 ≤ base_from, base_to ≤ 16`，且 `num_from` 是有效的 `base_from` 进制非负整数（长度 ≤ 10）。

🔄 转换流程图

整个转换过程分为三步：**源进制 → 十进制 → 目标进制**。

> 💡 关键点： `Dec2Base` 生成的是**低位在前**的字符串（例如 10 转二进制得到 `"0101"`），必须通过 `Reverse` 得到 `"1010"`。

🧪 测试用例（请确保你的程序能通过全部）

#	输入（base_from num_from base_to）	期望输出	说明
1	`10 123 2`	`1111011`	十进制转二进制
2	`2 1111 10`	`15`	二进制转十进制
3	`16 1A 2`	`11010`	十六进制转二进制
4	`8 77 16`	`3F`	八进制转十六进制
5	`10 0 2`	`0`	**边界：零值**
6	`2 0 16`	`0`	零在任意进制都是 "0"
7	`16 F 10`	`15`	单字符输入
8	`3 222 10`	`26`	三进制验证（2×9 + 2×3 + 2 = 26）
9	`10 255 16`	`FF`	十进制转十六进制
10	`16 0 10`	`0`	再次验证零

> ✅ 建议：先手动计算几个例子，再调试程序。

⚠️ 已知缺陷与思考题（仅思考即可）

尽管上述代码“能工作”，但它仍存在若干**设计缺陷和安全隐患**。请思考以下问题：

1. **错误处理缺失**：如果输入字符串包含非法字符（如 `'G'`）或某位数字 ≥ 当前进制（如在八进制中出现 `'9'`），程序会如何表现？应如何改进？
2. **数值溢出风险**：当前使用 `int` 存储十进制值。若输入是 16 位十六进制数（如 `"FFFFFFFF"`），会远超 `int` 范围（通常最大约 21 亿）。如何避免？
3. **缓冲区安全**： `num_to` 数组大小为 11，是否足够？例如，将 2 进制 `"1111111111"`（10 个 1）转为 16 进制只需 3 字符，但反过来呢？是否存在写越界风险？

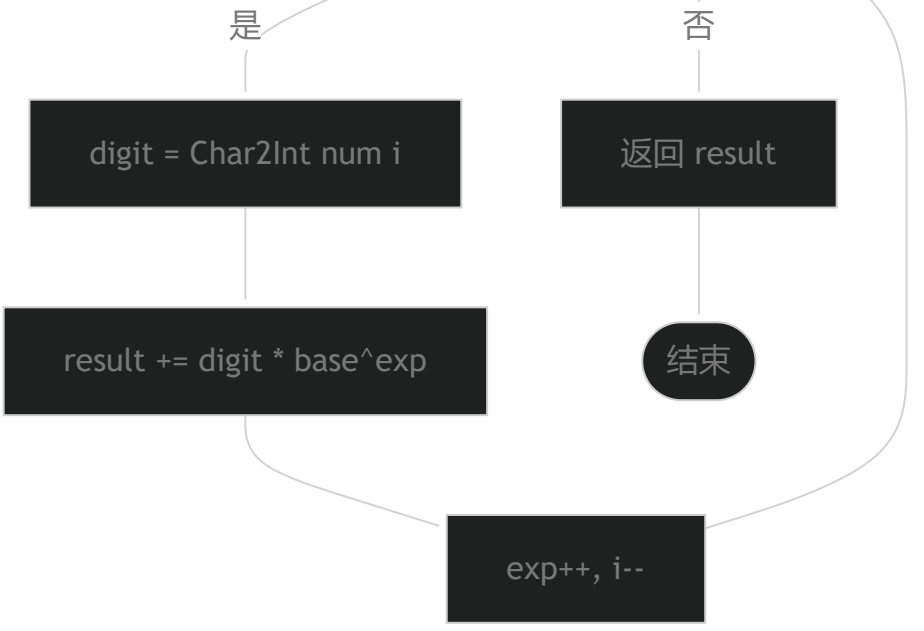
> 💡 提示：健壮的程序不仅要“正确”，还要“安全”和“容错”。

📎 提交要求

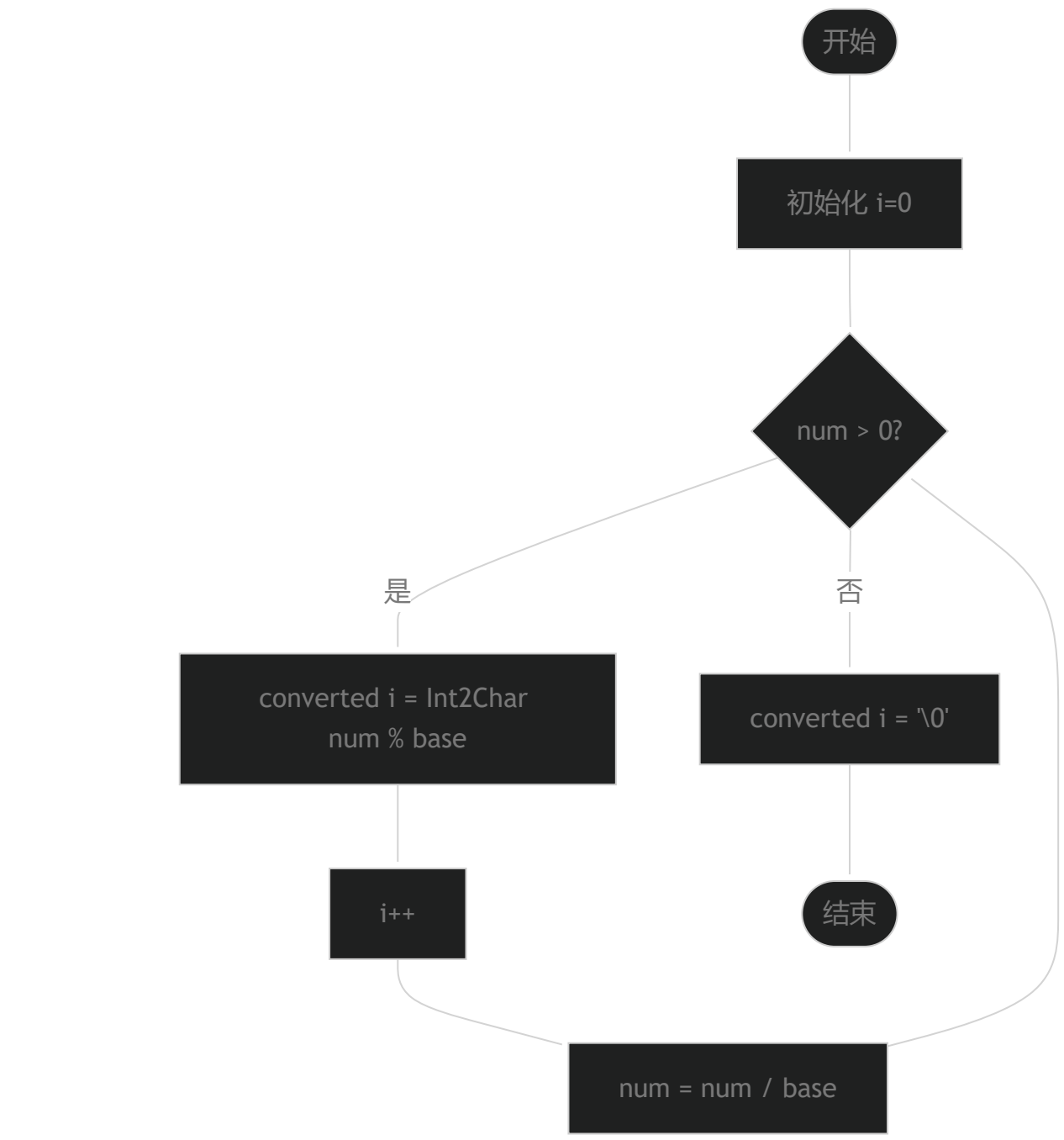
- 提交代码和所有测试例的运行截图。

💡 提示（尽量先自己思考，若毫无思路再参考以下流程图）

Base2Dec 函数流程图



Dec2Base 函数流程图


开始
初始化 i=0
num > 0?
是
否
converted i = Int2Char
num % base
converted i = '\0'
i++
结束
num = num / base