**1. 确定任务的变体**

* **步骤**: 从ISU号码中提取第三位和第五位数字组合。
* **示例**: 如果ISU号码是123456，那么变体是35。

**2. 选择消息集**

* **步骤**: 根据你选定的变体，选择4条消息，每条消息都应该是一个7位字符的代码。

**3. 构建经典的哈明码(7;4)解码方案**

* **步骤**: 绘制哈明码的解码结构。可以使用流程图工具（如Lucidchart或Draw.io）来创建此图。

**4. 错误检测**

* **步骤**: 根据所选的4条消息，检查每条消息是否有错误。可以使用哈明码的检测方法，找出错误位并修正。
* **示例**: 如果接收到的消息是“1101101”，并且发现错误位，你需要记录下来并给出修正后的消息。

**5. 选择单条消息**

* **步骤**: 根据变体选择一条11位的消息。

**6. 构建经典的哈明码(15;11)解码方案**

* **步骤**: 绘制哈明码(15;11)的解码结构，使用合适的图形工具。

**7. 错误检测**

* **步骤**: 对于所选的单条消息，使用哈明码的检测方法检查是否有错误并修正。

**8. 计算信息位和冗余位**

* **步骤**:
  + 将5个任务的变体号码相加。
  + 将结果乘以4，作为传输消息的信息位数。
  + 根据信息位数计算最小的冗余位数和冗余率。

**9. 额外任务**

* **步骤**: 编写程序分析输入的7位哈明码，检测错误并输出修正后的消息。你可以使用Python或其他编程语言来实现这一点

def hamming\_decode(received):

# 计算奇偶校验位

parity\_positions = [1, 2, 4]

error\_position = 0

for p in parity\_positions:

parity = 0

for i in range(1, len(received) + 1):

if i & p:

parity ^= int(received[i - 1])

if parity != 0:

error\_position += p

if error\_position:

print(f"Error found at position: {error\_position}")

# 纠正错误

received[error\_position - 1] = '1' if received[error\_position - 1] == '0' else '0'

# 提取信息位

message = received[2] + received[4] + received[5] + received[6]

return message, error\_position

# 输入示例

received\_code = '1101101' # 7位输入

corrected\_message, error\_pos = hamming\_decode(list(received\_code))

print(f"Corrected message: {corrected\_message}, Error position: {error\_pos if error\_pos else 'No error'}")