哈爾濱工業大學（威海）

Harbin Institute of Technology, Weihai

信息安全程序设计实践报告

任务题目：学生成绩匹配和管理系统

班　　级：2106101

学　　号：2021211351

姓　　名：夏天怡

开课学期：2023 年　秋　季学期

哈尔滨工业大学（威海）计算机科学与技术学院

前 言

信息安全程序设计实践是基于自选项目的实践训练课。学生将综合利用程序设计语言、数据结构、数据库等方面的基本概念、原理、技术和方法，开展面向实际应用的复杂工程问题设计求解和对应系统软件开发两大方面的实践。

通过本课程的学习、训练和实践，引导学生熟练掌握问题设计求解和软件编程开发的一般过程、相关技术、方法和途径；训练综合运用所学的理论知识和方法独立分析和解决问题，提高问题分析、问题求解和软件开发能力；培养学生能够针对实际问题，选择适当的数据结构或者数据库、设计有效算法，提高程序设计的能力和编码质量；训练和学会用系统的观点和软件开发一般规范进行软件设计开发，培养信息安全软件工作者所应具备的科学工作方法和作风，提高工程素质 。

本课程要求学生 通过一定的调研来自行结合实际应用需求来选题，并由任课教师来对学生选题做筛选评定。要求所设计开发的软件具有一定的实用性和系统完整性，要有较友好的图形交互操作界面，并对输入数据有较强的完整性约束，要以用户需求作为出发点来设计软件界面和功能模块。本课程主要教学环节包括：学生自选任务、开题检查、 软件验收、任务报告撰写提交等。

1.选题背景与意义

（描述选题的背景、所针对的具体实际问题及任务所体现的实用性价值等）

计划设计一款能够辅助老师教学活动的学生成绩匹配和管理系统，可以通过相应的方式获得网络平台学生测试成绩的 Excel 表格，这些散乱的数据表往往需要老师手动处理和管理，这种方式既费时又容易出错。因此，开发一款学生成绩匹配和管理系统成为迫切需求。系统应该具备多种功能，包括数据管理、异常检测、成绩分析等，同时支持灵活的扩展功能，以应对不同的教学需求。可以将该软件应用于现实中的很多场景，如在线学习平台的成绩统计和分析，如雨课堂、慕课等。

由于这个程序使用了 MD5 算法以及密码学中的一些哈希函数构造和和简单的移位密码的操作，让这个程序覆盖的知识面更加宽广了一些，本来想要更深层次的结合一些密码学的知识，可是由于程序本身的局限性没能做到，这里略微感到一些遗憾。

这个软件的优势在于能够使用真实的实验数据作为测试样例，可以直接在实际情况下进行应用，从而直接满足了用户的需求，而不需要进行很多复杂的检验和更改。

2.需求分析

（根据任务选题的要求，充分地分析和理解问题，明确用户要求做什么？运行环境要求、图形操作界面要求等）

首先用户要求获得一个可以帮助快速处理 Excel 成绩表格的可视化程序，这个程序应当拥有以下的几个功能：

1.能够自己读取判断并识别出总人数名单，并将总人数名单存储在自己的数据结构中。

2.能够将成绩单中的学生成绩与名单的那个学生姓名进行匹配，将每个成绩对应到每个人名单的姓名旁边，并且没有成绩的同学，成绩要进行补零。

3.程序需要能够读取两个完整的 Excel 表格，而不在读取文件过程中产生错误冲突，同时满足多次的运行原则，不能运行一次后再无法正常运行。

用户的要求是该程序能够自己帮助快速的处理 Excel 成绩的文件。完整性约束条件来自于完全真实的现实样例，对于任何不合规范的实验数据输入，必须要按照现实情况进行有针对性的解决。

运行环境主要为使用了 Python 环境，然后图形界面操作运用了 Python 的 tkinker 图形库进行了可视化设计，GUI 界面要求大方简洁，满足客户的日常使用需要。

3.系统主要功能设计

（根据需求分析来详细设计软件系统的主要功能模块，要求画出功能模块图，含子功能模块图，并给出详细文字描述）

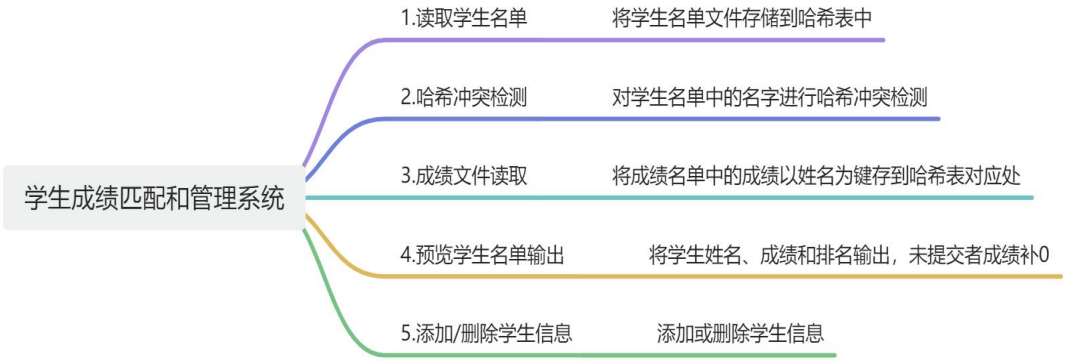

图 3-1 主要功能示意

软件首先会读取学生名单，然后将这些名单存储到一个哈希表中。它会检查是否有哈希冲突，接下来，软件会读取学生成绩文件，并将这些成绩对应到之前创建的哈希表中。如果有学生没有提交小测的成绩，系统将自动将这些成绩补零。然后，软件会输出每个学生的姓名以及对应的成绩和排名（利用插入排序），相同成绩排名一致。同时支持老师对学生信息进行添加和删除，完成所有功能后退出程序。

4.核心算法设计与分析

（根据软件功能设计，概述所用到的主要数据结构或者数据库等技术；用伪代码或程序流程图的形式来详细描述核心算法的功能及过程，并定性分析其时间、空间复杂度）

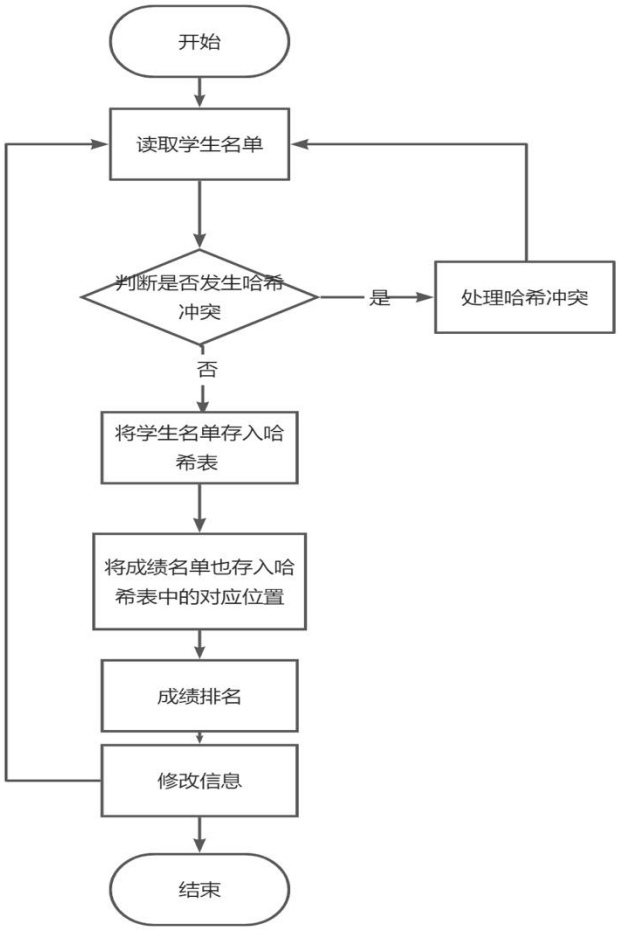

图 4-1 程序流程图

软件用了队列和哈希表两种数据结构。哈希表储存了学生姓名及成绩，队列则按顺序输出学生姓名和成绩。

核心算法是读取名单文件中的学生姓名，通过 MD5 加密转成 16 位 MD5 码，再转化成纯数字形式。使用取 Ascii 码值并对质数取余的方式得到小于 10 万的哈希值。在 100 以内学生情况下，哈希冲突约为千分之一，适用于实际应用。

针对哈希表，不发生哈希冲突的情况下，查找时间复杂度为 O（1），可直接通过哈希值找到对应位置。哈希表以 10 万大小数组开辟空间，故空间复杂度也为 O（1）。

对于成绩排序，使用插入排序，时间复杂度为 O（n^2）。

5. 系统核心模块实现

（给出编程语言、开发环境及支撑软件、软件系统架构；给出主要代码以及核心算法对应的关键函数定义代码，包括输入参数的含义、函数输出结果等；核心函数的实现代码段及注解等；主要功能界面截图及对应文字概述等）

图 5-1 软件架构

主函数：

```
if __name__ == "__main__":
    win = tk.Tk()
    win.geometry("500x400")  # 界面大小，中间是小写的 x
    win.title("学生成绩匹配和管理系统")
    # 加载图片
    image = Image.open("pic.gif")  # 替换为你的图片路径
    photo = ImageTk.PhotoImage(image)

    # 创建标签并设置图片
    label = tk.Label(win, image=photo)
    label.place(x=0, y=100, relwidth=1, relheight=1,anchor='nw')  # 使图片铺满整个窗口
    ft1 = tkFont.Font(family='Fixdsys', size=20, weight=tkFont.BOLD)
    ft2 = tkFont.Font(family='Fixdsys', size=10, weight=tkFont.BOLD)
    ft3 = tkFont.Font(family='Fixdsys', size=25, weight=tkFont.BOLD)
    ft4 = tkFont.Font(family='Fixdsys', size=15, weight=tkFont.BOLD)
    Label_first = tk.Label(win)
    Label_second = tk.Label(win)
    print("请输入要储存的学生名单文件名")
    print("请输入要打开的成绩文件名")
    s2="请依次输入学生名单的文件名和学生成绩单，如 AA.xlsx 和 BB.xlsx"
    var1 = StringVar()
    var2 = StringVar()
    Label_first=tk.Label(win,text=s2,bg='LightGreen',
fg='Tomato',font=ft2).place(x=55, y=0)
    Label_second = tk.Label(win, text="学生名单文件名", bg='LightGreen',
fg='Maroon', justify='left').place(x=50, y=50)
```

```
   Label_third = tk.Label(win, text="学生成绩单文件名", bg='LightGreen', fg='Maroon', justify='left').place(x=50, y=100)
   text1 = tk.Entry(win, width=35, bg='White', textvariable=var1).place(x=170, y=50)
   text2 = tk.Entry(win, width=35, bg='White', textvariable=var2).place(x=170, y=100)
   button_First = tk.Button(win, width=5, text='Run!', fg='blue', command=lambda: menu(var1.get(),var2.get())).place(x=220, y=150)
   win.mainloop()
```

功能 1：读取学生名单

```
def name(s):
    m = 0  # 行坐标
    df3 = pd.read_excel(s)  # 打开文件

    while 1:

        try:
            namelist = df3.iat[m, 0]  # 姓名顶在（0，0）开始
        except Exception as e:  # 这里解决了一个文件指针的越界异常，一旦越界就返回菜单
            break

        if namelist == '':
            break

        q1.put(namelist)
        m = m + 1  # 进到下一行
    wr()

def  wr():
        f1  =  open("name.in",  "w")
        while  1:
                if  q1.empty():    #检验队列是否为空
                        break
                #print(q1.get())    #不为空输出队首且将队首删除
                f1.write(str(q1.get()))
                f1.write('\n')
        f1.close()
```

功能 2：哈希冲突检测

```
def conflict(s,s1):
   m = 0  # 行坐标
   flag=0
```

```
    # 行号和列号必须从 0 开始，获取指定单元格的内容

    df1 = pd.read_excel(s)  # 打开文件

    while 1:
        # 行号和列号必须从 0 开始，获取指定单元格的内容
        f3 = open("conflict.in", "w", encoding='UTF-8')  # 和上面的 df1 冲突，很离谱，无法解决
        try:
            namelist = df1.iat[m, 0]  #姓名顶在（0，0）开始
        except Exception as e:   #这里解决了一个文件指针的越界异常，一旦越界就返回菜单
            print(s,s1)
            break

        if namelist == '':
            break
        m = m + 1  #进到下一行
        s2= ''  # 储存字符串
        s3 = hashlib.md5(namelist.encode(encoding='UTF-8')).hexdigest()  # md5 将中文转字母+数字
        # print(s)   #测试输出用
        for ch in s3:  # 第一个实例
            s2 = s2 + str(ord(ch) - 40)  # 移位加密，0 的 ascii 码是 48
        # print(s1)   #测试输出用
        pos = int(s2) % 9973  # hash 表的质数取余法 9973 为一个接近 10000 的大质数
        #print(namelist,pos)  # 测试哈希值用
        if b[pos] == 0:
            b[pos] = 1
        else:
            print(namelist, "出现哈希冲突")
            f3.write(namelist)
            f3.write("出现哈希冲突")
    f3.close()
    window2 = tk.Toplevel(win)
    window2.configure(bg='LightYellow')
    window2.title("冲突检测")
    window2.geometry("500x250")
    #Label(window2, text="请选择要执行的功能:", font=ft4).place(x=50, y=0)
    f2 = open("conflict.in", "r",encoding='UTF-8')
    code = f2.read()
    f2.close()
    text = tk.Text(window2)
    text.insert('end', code)
    text.place(x=0, y=60)
```

功能 3:成绩文件读取

```
def read(s,s1,m,y1,y2):
    # 行号和列号必须从 0 开始，获取指定单元格的内容
    df = pd.read_excel(s1)  # 打开文件
    while 1:
        try:
            name = df.iat[m, y1]  # 读取第 m 行第六列的值，这里不需要嵌套列表
            score = df.iat[m, y2] #读取第 m 行第零列的值，这里不需要嵌套列表
        except Exception as e:  #这里解决了一个文件指针的越界异常，一旦越界就返回菜单
            break

        if name == '':
            break
        hash(name, score)
        # print(name,score)
        m = m + 1

def  hash(name,score):
        s1=''          #储存字符串
         s=hashlib.md5(name.encode(encoding='UTF-8')).hexdigest()#md5 将中文转字母+数字
        #print(s)       #测试输出用
        for  ch  in  s:     #  第一个实例
                s1=s1+str(ord(ch)-40)       #移位加密，0 的 ascii 码是 48
        #print(s1)       #测试输出用
        pos=int(s1)%99991     #hash 表的质数取余法  9973 为一个接近 10000 的大质数
        print(name,pos)       #测试哈希值用
        if  a[pos]==0:
                a[pos]=score
        else:
                while  a[pos]!=0:
                        pos=pos+1
                a[pos]=score
                #print(name,"出现哈希冲突")
                #print("一旦出现哈希冲突，我们便认为这个算法是不够安全的，在实际应用中建议将冲突学生姓名修改成  "",  m,  "1“  便可以保证算法的安全性")
```

功能 4：预览学生名单输出

```
def output(s2,s3):
    m = 0  # 行坐标
    # 行号和列号必须从 0 开始，获取指定单元格的内容

    df1 = pd.read_excel(s2)  # 打开文件
```

```
    while 1:
        # 行号和列号必须从0开始，获取指定单元格的内容
        try:
            namelist = df1.iat[m, 0]
        except Exception as e:  #这里解决了一个文件指针的越界异常，一旦越界就返回菜单
            break
        if namelist == '':
            break
        m = m + 1  #进到下一行
        s1 = ''  # 储存字符串
        s = hashlib.md5(namelist.encode(encoding='UTF-8')).hexdigest()  # md5将中文转字母+数字
        # print(s)   #测试输出用
        for ch in s:  # 第一个实例
            s1 = s1 + str(ord(ch) - 40)  # 移位加密，0的ascii码是48
        pos = int(s1) % 9973  # hash表的质数取余法 9973为一个接近10000的大质数
        if a[pos] != -1:
            print(namelist,a[pos])  #哈希表删除时不能将该点元素删除，否则查找不到后面的冲突元素，一般置一个取不到的特殊符号，这里置-1
            q.put(a[pos])      #将成绩入队
            a[pos]=-1
        else:
            while a[pos]==-1:
                pos=pos+1
            print(namelist, a[pos])  # 哈希表删除时不能将该点元素删除，否则查找不到后面的冲突元素，一般置一个取不到的特殊符号，这里置-1
            q.put(a[pos])
            a[pos] = -1
    output1(s)

def output1(s):
    score_sort()
    # Rest of your code for tkinter window...
    window3 = tk.Toplevel(win)
    window3.configure(bg='LightYellow')
    window3.title("成绩预览")
    window3.geometry("800x550")

    f_out = open("sorted_scores.out", "r", encoding='UTF-8')
    code1 = f_out.read()
    f_out.close()

    text1 = tk.Text(window3, width=100, height=30)
    text1.insert('end', code1)
```

```
    text1.place(x=10, y=60)
```

成绩排名（插入排序）：按分数从高到低降序，并且有同样分数排名相同，名次累加

```
def score_sort():
    print("test sort")
    f_score = open("score.in", "w")
    while 1:
        if q.empty():  #检验队列是否为空
            break
        #print(q.get())  #不为空输出队首且将队首删除
        f_score.write(str(q.get()))
        f_score.write('\n')
    f_score.close()
    f_score= open("score.in", "r", encoding='UTF-8')
    # code1=f1.read()
    scores =f_score.readlines()
    f_score.close()
    with open("name.in", "r", encoding='gbk') as f_name:  #open("score.in",
"r", encoding='UTF-8') as f_score:
        names = f_name.readlines()
        # scores = f_score.readlines()

    if len(names) != len(scores):
        print("Error: Number of names and scores don't match!")
        return

    # Combine names, scores, and indices into a list of tuples
    data = [(names[i].strip(), int(float(scores[i].strip()))) for i in
range(len(names))]

    # Implementing insertion sort and sort by scores
    for i in range(1, len(data)):
        key = data[i]
        j = i - 1
        while j >= 0 and data[j][1] < key[1]:
            data[j + 1] = data[j]
            j -= 1
        data[j + 1] = key

    # Assign ranks based on sorted order
    ranks = [1] * len(data)
    for i in range(1, len(data)):
        if data[i][1] == data[i - 1][1]:
            ranks[i] = ranks[i - 1]
        else:
            ranks[i] = ranks[i - 1] + 1
```

```
    # Combine sorted data with ranks
    #result = [f"姓名：{data[i][0]}，成绩：{data[i][1]}，排名：{ranks[i]}" for i in range(len(data))]
    # Combine sorted data with ranks, keeping only the first occurrence of each name
    result = []
    seen_names = set()
    for i in range(len(data)):
        if data[i][0] not in seen_names:
            result.append(f"姓名：{data[i][0]}，成绩：{data[i][1]}，排名：{ranks[i]}")
            seen_names.add(data[i][0])
    # Save sorted result to a file
    with open("sorted_scores.out", "w", encoding='UTF-8') as f_out:
        f_out.write('\n'.join(result))
```

功能 5：添加/删除学生信息

```
def edit(s,s1):
   print(s1)
   # copy_sheet(s1,'Sheet1','e.xlsx','Sheet1')
   # tt='e.xlsx'
   wb = load_workbook(s1)  # 修改成绩文件
   wb1= load_workbook(s)   # 修改名单文件
   print('1111')
   def submit_info(wb,wb1,s1,s):
       # 获取输入框中的信息
       name = entry_name.get()
       student_id = entry_id.get()
       score = entry_score.get()
       flag = int(entry_flag.get())
       print(flag)
       # 打开 Excel 文件
       # wb = load_workbook('e.xlsx')
       ws = wb.active
       ws1= wb1.active

       if flag==1:
       # 将信息添加到 Excel 的最后一行
           last_row = ws.max_row + 1
           ws.cell(row=last_row, column=2).value = name
           ws.cell(row=last_row, column=3).value = student_id
           ws.cell(row=last_row, column=4).value = score

           last_row1=ws1.max_row+1
           ws1.cell(row=last_row, column=1).value = name
       elif flag==2:
```

```
            print("here")
            for   row   in   ws.iter_rows(min_row=1,   min_col=2,max_col=2,
max_row=ws.max_row):
                for cell in row:
                    if cell.value == name:
                        ws.delete_rows(cell.row, 1)  # 删除找到的行
            for   row   in   ws1.iter_rows(min_row=1,   min_col=1,max_col=1,
max_row=ws1.max_row):
                for cell in row:
                    if cell.value == name:
                        ws1.delete_rows(cell.row, 1)  # 删除找到的行
        # 保存并关闭 Excel 文件
        wb.save(s1)
        wb.close()
        wb1.save(s)
        wb1.close()

    window_edit = tk.Toplevel(win)
    window_edit.configure(bg='LightBlue')
    window_edit.title("学生信息修改界面")
    window_edit.geometry("600x500")

    Label(window_edit, text="请输入以下信息:", bg='LightYellow', fg='Tomato',
font=ft4).place(x=50, y=0)

    # 创建输入框和标签
    Label(window_edit, text="添加 or 删除信息：输入 1or2", bg='LightGreen',
fg='Maroon').place(x=400, y=70)
    entry_flag = tk.Entry(window_edit)
    entry_flag.place(x=400, y=100)

    Label(window_edit, text="姓名:", bg='LightGreen', fg='Maroon').place(x=50,
y=50)
    entry_name = tk.Entry(window_edit)
    entry_name.place(x=150, y=50)

    Label(window_edit, text="学号:", bg='LightGreen', fg='Maroon').place(x=50,
y=100)
    entry_id = tk.Entry(window_edit)
    entry_id.place(x=150, y=100)

    Label(window_edit, text="成绩:", bg='LightGreen', fg='Maroon').place(x=50,
y=150)
    entry_score = tk.Entry(window_edit)
    entry_score.place(x=150, y=150)
```

```
    # 提交按钮
    submit_button = tk.Button(window_edit, width=20, text="提交", fg='black',
font=ft4, command=lambda: submit_info(wb,wb1,s1,s))
    submit_button.place(x=200, y=200)
```

主要功能界面截图：

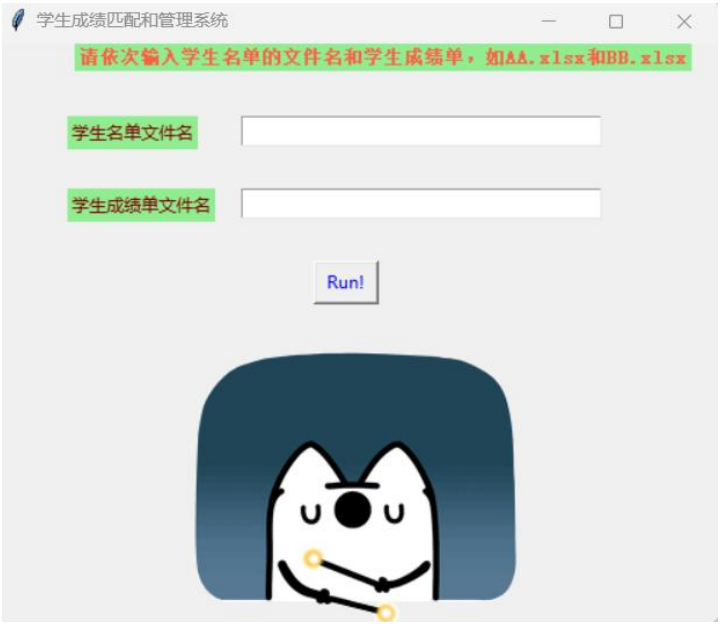

图 5-1 主页面

软件首先会读取学生名单，并将这些名单存储到一个哈希表中。通过哈希表，它会尝试减少重复的姓名。然后，程序会进行哈希冲突的检测，即使哈希表可以减少重复，但还需要确保没有发生哈希冲突。在实际情况下，百分之百的正确率难以保证，但这个方法可以大幅降低重复率。

接下来，软件会读取学生成绩文件，并将这些成绩对应到之前创建的哈希表中。如果有学生没有提交小测的成绩，系统将自动将这些成绩补零。最后，软件会输出每个学生的姓名以及对应的成绩。

图 5-2 菜单界面

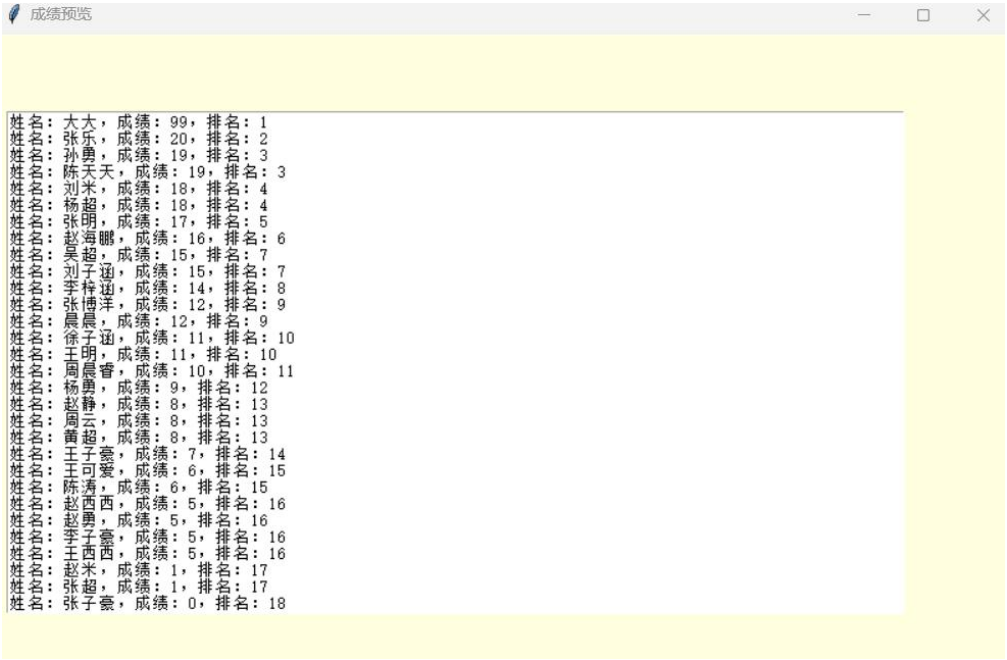

图 5-3 成绩预览

输出学生姓名和成绩，提交失败者成绩为 0.

图 5-4 学生信息修改

添加学生信息输入 1，删除输入 2，添加和删除信息都是姓名匹配。程序会对源 xlsx 文件进行修改。

6. 调试分析记录

（软件开发调试过程中遇到的问题及解决过程；核心算法的运行时间和所需内存空间的量化测定；符合实际情况的数据测试，算法及功能的改进设想等）

在软件开发过程中，主要难点在于哈希表的构建和哈希函数的定义。哈希函数的核心算法使用了 MD5 对学生姓名进行处理，并将 MD5 码转化为 Ascii 码，再进行移位加密，最终得到纯数字形式，然后应用质数取余法完成哈希表的建立。

关键的查找算法，在没有哈希冲突的情况下，确实可以在 O(1)时间复杂度内查找数据，同时哈希表所占的内存空间是预先确定的常量，空间复杂度为 O(1)，这里哈希表的内存空间大小设为 10 万。

这款软件在实际的成绩录取表格中进行了数据测试，得到了准确的结果，显著提升了用户处理成绩表格的效率。在改进算法方面，考虑将对姓名的 MD5 处理改为对学号取余，可能极大降低哈希冲突的概率，同时避免了重名情况对成绩录取造成的困难。

这种优化方案有望降低重复率，提高查重的准确性，同时优化哈希函数，使其更符合实际情况，能够更有效地处理学生数据。

在成绩排序时，采用插入排序算法，但是需要额外考虑成绩相同的情况，为此，使用新的 result 数组，对排序好的数组重新进行遍历，存入 result。

```
result = []
seen_names = set()
for i in range(len(data)):
   if data[i][0] not in seen_names:
       result.append(f"姓名:{data[i][0]},成绩:{data[i][1]},排名:{ranks[i]}")
       seen_names.add(data[i][0])
```

关于数据信息在不同函数传递，采用读写.in/.out 文件的方式。

在修改学生信息部分，采用对 xlsx 文件直接修改然后重新读取的方式，这样可能有些耗时，且程序安全性降低。但是时间紧迫，且直接修改也不是很容易，我就还是选择了这样。主要是借助了 from openpyxl import load_workbook, Workbook。

7. 运行结果与分析

（要有多组正确的实际测试数据，并给出相应运行结果截图，并对多组结果进行比较分析）

真实姓名
王可爱
刘米
徐子涵
赵西西
张子豪
王子豪
吴超
赵勇
周子涵
赵静
孙勇
赵海鹏
周云
王明
张乐
周晨睿
杨勇
张博洋
李子豪
陈涛
王西西
李梓涵
刘子涵
黄超
陈天天
赵米
杨超
王强
张明
张超
晨晨
大大

学生昵称	真实姓名	学号	成绩
Student_1	王可爱	2021201422	6
Student_2	刘米	2021210164	18
Student_3	徐子涵	2021212428	11
Student_4	赵西西	2021210557	5
Student_5	张子豪	2021212759	0
Student_6	王子豪	2021201679	7
Student_7	吴超	2021211864	15
Student_8	赵勇	2021211870	5

Student_9	周子涵	2021211286	0
Student_10	赵静	2021212503	8
Student_11	孙勇	2021202601	19
Student_12	赵海鹏	2021212288	16
Student_13	周云	2021202916	8
Student_14	王明	2021211937	11
Student_15	张乐	2021211918	20
Student_16	周晨睿	2021212785	10
Student_17	杨勇	2021201258	9
Student_18	张博洋	2021200230	12
Student_19	李子豪	2021211822	5
Student_20	陈涛	2021210602	6
Student_21	王西西	2021212234	5
Student_22	李梓涵	2021212928	14
Student_23	刘子涵	2021202520	15
Student_24	黄超	2021201455	8
Student_25	陈天天	2021211421	19
Student_26	赵米	2021200103	1
Student_27	杨超	2021201214	18
Student_28	王强	2021212776	0
Student_29	张明	2021212819	17
Student_30	张超	2021202470	1
Student_31	晨晨	2021211301	12
	大大	2021201002	99
	大大	2021201002	99
	大大	2021201002	99

表 7-1，2 学生名单和学生成绩单

未检测到哈希冲突：

图 7-1 冲突检测

输出成绩排名：

图 7-2：成绩情况显示

修改学生信息：

删除王西西：成功删除

添加：啊啊啊 2021 98

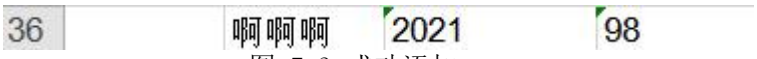

图 7-3 成功添加

更换数据集，添加重复学生啊啊啊：

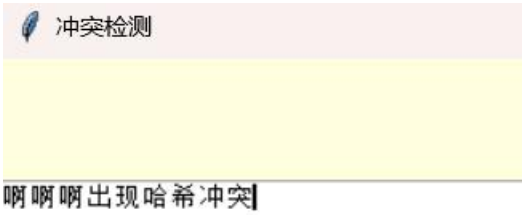

图 7-4 更换数据集，产生哈希冲突

8. 总结（收获与体会）

（如实撰写课程任务完成过程的收获和体会以及遇到问题的思考、程序调试能力的培养提升等相关内容；要求不少于 500 字，严禁雷同）

老师上课经常会用到各种在线答题平台，如何处理这些成绩数据是我一直好奇的（虽然我知道很多平台的算法已经很完备了），正好借着这次机会，我从 0 学习 python 画 ui 界面，然后利用数据结构相关知识自己实现了一个简单的成绩管理系统。

整个哈希存储和哈希冲突解决的是几个问题：1.更快更方便的存储姓名和成绩信息 2.如何解决有同学提交失败的问题 3.如何解决同学重复提交的问题。

我使用的是哈希表的存储结构,所以我将学生的姓名取了 MD5 值,接着再去取 Ascii 码，转化成纯数字的形式，然后运用了哈希表中的质数取余法和线性探测法完成了哈希表的构造。

在读取学生成绩名单之前，我首先将哈希表中所有学生的初始成绩置为了 0，然后再从成绩文件中读取学生成绩，如果有则录入，没有的话则继续为 0，最后读取学生的总名单文件，将其对应的成绩进行输出。

在对 Excel 表格进行操作的时候，有一个文件指针指到尾部的函数总是难以找到，最终我自己想了想使用了一个抛出异常的处理机制，一旦读到文件尾，它就会抛出一个异常，读到异常以后这个函数就被终止了，从而完美的解决了这个问题。

在可视化程序的设计当中,我发现用 python 进行可视化界面设计要比之前我用 mfc 和 qt 方便很多，这也是我这次使用它的原因。无论是添加还是修改界面，以及按键函数都很方便。

对于使用 python 进行 excel 表格的读取和修改的方法我也有了更多的了解。另外，由于是对数据进行操作，这样每个细节的处理都很重要。如符号、重复数据、中文汉字等特殊情况都是需要考虑的。

在这次的课程设计中，首先我提升了我独自完成一个项目的程序编写能力以及对于 bug 的纠错心得。养成了遇到很多不懂的功能，及时进行网上查找相关的函数功能进行实现的好习惯。同时在 debug 的过程中磨练了自己处变不惊以及有耐心的好习惯，以及程序一点一点推进,遇到一个 bug 就解决一个 bug,而不是在全部编写完以后一起 debug，这是我觉得尤为重要的一点，也是我学到的一个非常有意义的习惯。