



INDIVIDUAL ASSIGNMENT

SYSTEM DOCUMENTATION

CT018-3-1-ICP

INTRODUCTION TO C PROGRAMMING

Student Name: LEE ZHI XUAN

Student ID: TP065525

Intake Code: APD1F2206CE

Lecturer Name: MS. SUPRIYA SINGH

Hand In Date: 27 March 2023

Weightage: 50%

1.0 Table of Contents

1.0 Table of Contents	2
2.0 Introduction.....	5
2.0 Assumptions.....	6
2.1 Limitations	8
3.0 Design of the Program (Pseudocode)	9
3.1 Main file.....	9
3.1.1 Main function.....	9
3.1.2 Title screen function.....	10
3.1.3 Staff info structure	10
3.1.4 Login validity checker	10
3.1.5 Main menu function.....	11
3.1.6 Login menu function.....	12
3.1.7 Import current system time function.....	14
3.1.8 Display user info function.....	14
3.1.9 Read staff info from text file function	15
3.1.10 Write staff info into text file function	16
3.1.11 Read leave info from text file function	17
3.1.12 Write leave info into text file function.....	17
3.2 Staff file	18
3.2.1 Staff menu function.....	18
3.2.2 Apply leave function.....	19
3.2.3 Day validity checker function.....	22
3.2.4 Cancel leave function.....	23

3.2.5 Leave status function	26
3.2.6 Leave list function.....	26
3.3 Supervisor file	28
3.3.1 Supervisor menu function	28
3.3.2 Application management function	29
3.3.3 Application processing menu function.....	31
3.3.4 Date lookup function.....	34
3.4 Admin file	36
3.4.1 Admin menu function	36
3.4.2 Staff information lookup function	38
3.4.3 Generate monthly report function.....	40
3.4.4 Edit leave balance function.....	43
3.4.5 Add staff information function.....	46
4.0 Program Source Code Concepts and Explanations.....	49
4.1 File Handling	49
4.2 Leave information file data sorting.....	52
4.3 Converting character data type into integer data type	54
4.4 Function with arguments.....	55
4.5 C structs	56
4.6 ANSI colour codes	57
5.0 Sample Input/Output with Explanations.....	58
5.1 Starting menu and login	58
5.2 Staff role.....	61
5.3 Leave application menu	62
5.4 Leave cancellation menu.....	65

5.5 Leave status and information menu	68
5.6 Supervisor role	69
5.7 Application management menu.....	70
5.8 Date lookup menu	72
5.9 Admin role	73
5.10 Add staff.....	74
5.11 Edit leave balance	76
5.12 View staff information	78
5.13 Generate monthly report	80
6.0 Conclusion	82
7.0 References.....	83
8.0 Appendix	84
8.1 Backup Source Code Download	84

2.0 Introduction

Over the past couple of decades, efficiency and convenience has become the hallmarks of contemporary society. Emphasis on these qualities enabled the rapid advancement of humanity by streamlining and cutting down processes to allow resources to be diverted to more productive sectors. One such example is by offloading the traditionally laborious task of handling, keeping track, and managing large amounts of data to automated computer systems.

The objective of this assignment is to develop an employee leave application and management system in the C programming language. The simulated customer for this assignment is Asia Pacific University, a well know private university located in Bukit Jalil, Kuala Lumpur with more than 500 staff members. The system is designed to simplify and streamline the process of managing employee leave requests and approvals. By automating and digitizing the leave handling processes, we aim to cut down on human labor and in turn, eliminate human error.

The system developed is capable of handling employee leave applications, cancellations, as well as check up on their leave status. On the management and administrative side, supervisors can approve or reject leave applications of staff from their department and view names of employees on leave on a particular date. Admins reserve the power to add staff members, edit staff leave balances, lookup the information of staff members, as well as generate a monthly staff leave report.

2.0 Assumptions

From the assignment specifications, it is assumed that all employees are categorized into 4 departments which are: administrative, management, academic, and technical. In addition, there are also 3 ranks which grants the employee different levels of permission in the leave management system. Each employee is assigned one of the 3 ranks and assigned to one of the departments.

The administrative department consist of all university administrators and at least one supervisor who oversees the leave applications of the admins. The other 3 departments each have at least one supervisor that handles the leave applications of their respective departments. The departments other than the administrative will not have admins and will only consist of employees of staff and supervisor ranks. The supervisors are the heads of their departments and are one step below that of an admin except for admin supervisors.

The assumed hierarchy structure can be better described in the figure below:

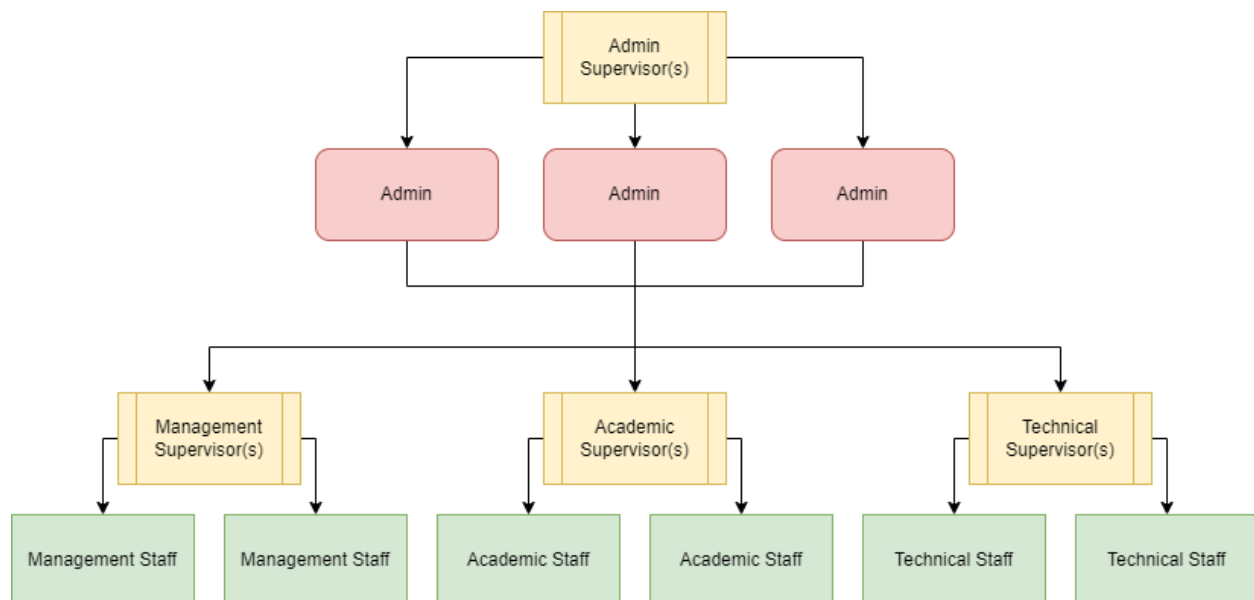


Figure 2.1: *Employee hierarchy*

The functions of the staff rank are universal. Meaning that employees with the admin or supervisor rank also has access to basic leave functions such as application, cancellation, and review of leave status through their respective admin and supervisor accounts.

By default, the system comes with 4 supervisor accounts, one for each department. The administrative department comes with 3 admin accounts, and the remaining 3 departments has 2 staff members each. Since the program requires the user to log in to access any of its functions, there's an additional "default" account with the admin rank in case other accounts are inaccessible. The default leave balances for all employees are 20 days per year.

In addition, even though the system can detect the current date, month, and year, the updating of staff leave balances for a new year is kept as a manual operation for admins. This is to leave open the possibility that the company's internal leave year may not align with the calendar year.

Moreover, leaves are separated into 5 categories, namely: annual, compassionate, emergency, maternity, and medical. However, all 5 categories share the same leave balance pool to cut down on complexity.

Lastly, the program is split into 5 different .cpp files, one for each staff rank, one for aesthetic functions, and one main function. Importing all 5 files and running the Source.cpp (main) file should start the program as intended.

2.1 Limitations

Unfortunately, there are several limitations in this program that were unable to be addressed during the limited timeframe of this assignment. The first is that the program does not check for whether the applied leave day is a weekday, weekend, or public holiday. It is possible that a staff may apply for leave on a non-workday. Thus, it is assumed that the leave applicator has an external calendar reference and is aware of the situation of the day of which the leave is applied.

Next, the program only allows users to apply for leave one day at a time. If the user wishes to apply for multiple consecutive leave days, they will have to do so individually for every single day. This decision was made to cut down on the complexity of the program and its user interface.

Moreover, the program also does not check for whether the user has already applied for leave on the selected leave date. Meaning that is it possible that a user may apply for leave twice on the same date. A solution to this limitation has already been identified but were unable to be implemented due to time constrains. For now, it is suggested that the user keep track of their leave applications from the leave information menu and refer to it before making a new application.

In addition, the program is also only capable of adding new staff, but is unable to easily remove them once added. Users can still be removed, but it is a manual operation by removing the staff's information from the text file. For now, this issue is partially rectified by the presence of a confirmation layer where the system will prompt the user to double confirm their decision when adding a new staff member into the system.

Lastly, the program also lacks the ability to edit the leave balance of individual staff members through the program. However, this can still be accomplished by editing the text files themselves.

3.0 Design of the Program (Pseudocode)

3.1 Main file

3.1.1 Main function

BEGIN

 DEFINE max_staff_count as 1000

 DEFINE max_username_length as 50

 DEFINE max_staff_id_length as 50

 DEFINE max_password_length as 50

 DEFINE max_balance as 100

 DEFINE inverted_balance as inverted balance of staff index

 DECLARE global staff_count, staff_select, and date variables

 CALL read_info_file()

 CALL read_leave_file()

 CALL time()

 CALL main_menu()

 RETURN

END

3.1.2 Title screen function

DECLARE title()

PRINT "=====

PRINT "APU – LEAVE APPLICATION AND MANAGEMENT SYSTEM"

PRINT "=====

END OF FUNCTION

3.1.3 Staff info structure

DECLARE STRUCTURE staff_details

DECLARE username[max_username_length]

DECLARE staff_id[max_staff_id_length]

DECLARE password[max_password_length]

DECLARE perm_level, balance, inverted_balance, department

DECLARE type[max_balance]

DECLARE day[max_balance]

DECLARE month[max_balance]

DECLARE year[max_balance]

DECLARE status[max_balance]

DECLARE staff_details structure as staff[max_staff_count]

3.1.4 Login validity checker

DECLARE loginValid(staff_details, staff_id[], password[])

FOR (i = 0 to staff_count) STEP 1

```
        IF (staff[i].staff_id same as staff_id)

            RETURN true

        ENDIF

    ENDFOR

    RETURN false

END OF FUNCTION
```

3.1.5 Main menu function

```
DECLARE main_menu()

    DOWHILE (selection is not exit)

        SYSTEM clear screen

        CALL title()

        PRINT login menu texts

        PRINT "Selection"

        ACCEPT selection

        IF selection is 1

            CALL login()

        ELSE IF selection is e

            PRINT "Exiting..."

            CALL write_info_file()

            CALL write_leave_file()

            EXIT

        ELSE
```

PRINT "Invalid selection. Please try again."

ENDIF

ENDDO

END OF FUNCTION

3.1.6 Login menu function

DECLARE login()

DECLARE login_attempts = 3, flag, i

SYSTEM clear screen

CALL title()

PRINT menu header texts

DOWHILE (flag is 0)

DECLARE staff_id[max_staff_id_length], password[max_password_length]

PRINT "Staff id:"

ACCEPT staff_id

FOR (i = 0 to staff_count) STEP 1

IF (staff[i].staff_id same as staff_id)

staff_select = i

BREAK

ENDIF

ENDFOR

PRINT "Password:"

ACCEPT password

```
IF (CALL loginValid(staff, staff_id, password))

    flag = 1

    PRINT "Login successful."

    IF staff_perm is 1

        CALL staff_menu()

    ELSE IF staff_perm is 2

        CALL supervisor_menu()

    ELSE IF staff_perm is 3

        CALL admin_menu()

    ENDIF

    BREAK

ENDIF

IF (login_attempts more than 1)

    SYSTEM clear screen

    CALL title()

    PRINT remaining attempts

ELSE

    PRINT "Incorrect password"

    EXIT

ENDIF

ENDDO

END OF FUNCTION
```

3.1.7 Import current system time function

```
DECLARE time()

    IMPORT time using time function

    IF (error)

        PRINT "Failed to get local time!"

        RETURN 1

    ENDIF

    current_day = TIME day

    current_month = TIME month

    current_year = TIME year

    RETURN 0

END OF FUNCTION
```

3.1.8 Display user info function

```
DECLARE user_info()

    PRINT header texts

    PRINT staff_select username

    PRINT staff_select staff_id

    IF rank is 1

        PRINT "Staff"

    ELSE IF rank is 2

        PRINT "Supervisor"

    ELSE IF rank is 3
```

```
        PRINT "Admin"

    ENDIF

    IF dept is 1

        PRINT "Administrative"

    ELSE IF dept is 2

        PRINT "Management"

    ELSE IF dept is 3

        PRINT "Academic"

    ELSE IF dept is 4

        PRINT "Technical"

    ENDIF

    PRINT leave balance

END OF FUNCTION
```

3.1.9 Read staff info from text file function

```
DECLARE read_info_file()

    DECLARE i

    OPEN staff_info.txt as staffinfo

    IF (staffinfo is 0)

        PRINT "Failed to read from file!"

    ENDIF

    FOR (i = 0 to end of file) STEP 1
```

```
        READ staffinfo line staff[i].username, staff[i].password, staff[i].staff_id,  
        staff[i].perm_level, staff[i].balance, staff[i].inverted_balance, staff[i].dept  
  
    ENDFOR  
  
    staff_count = i - 1  
  
    CLOSE staffinfo  
  
END OF FUNCTION
```

3.1.10 Write staff info into text file function

```
DECLARE write_info_file()  
  
    OPEN staff_info.txt as staffinfo  
  
        IF (staffinfo is 0)  
  
            PRINT "Failed to write into file!"  
  
        ENDIF  
  
        FOR (i = 0 to staff_count) STEP 1  
  
            Write      staff[i].username,      staff[i].password,      staff[i].staff_id,  
            staff[i].perm_level, staff[i].balance, staff[i].inverted_balance, staff[i].dept  
            into staffinfo file  
  
        ENDFOR  
  
        CLOSE staffinfo  
  
END OF FUNCTION
```


3.1.11 Read leave info from text file function

```
DECLARE read_leave_info()

    DECLARE i, j

    OPEN leave_info.txt as leaveinfo

    IF (staffinfo is 0)

        PRINT "Failed to read from file!"

    ENDIF

    FOR (i = 0 to staff_count) STEP 1

        FOR (j = 0 to staff[i].inverted_balance) STEP 1

            READ leaveinfo staff[i].type[j], staff[i].day[j], staff[i].month[j],
            staff[i].year[j], staff[i].status[j]

        ENDFOR

    ENDFOR

    CLOSE leaveinfo

END OF FUNCTION
```

3.1.12 Write leave info into text file function

```
DECLARE write_leave_info()

    DECLARE i, j

    OPEN leave_info.txt as leaveinfo

    IF (staffinfo is 0)

        PRINT "Failed to write into file!"

    ENDIF
```

```
FOR (i = 0 to staff_count) STEP 1

    FOR (j = 0 to staff[i].inverted_balance) STEP 1

        WRITE staff[i].type[j], staff[i].day[j], staff[i].month[j], staff[i].year[j],
            staff[i].status[j] into leaveinfo

    ENDFOR

ENDFOR

CLOSE leaveinfo

END OF FUNCTION
```

3.2 Staff file

3.2.1 Staff menu function

```
DECLARE staff_menu()

DECLARE selection

DOWHILE (selection is not e)

    SYSTEM clear screen

    CALL title()

    PRINT menu items

    CALL user_info()

    PRINT menu options

    PRINT "Selection:"

    ACCEPT selection

    IF selection is 1

        CALL apply_leave()
```

```
        ELSE IF selection is 2

            CALL cancel_leave()

        ELSE IF selection is 3

            CALL leave_status()

        ELSE IF selection is b

            CALL main_menu()

        ELSE IF selection is e

            PRINT "Exiting..."

            CALL write_info_file()

            CALL write_leave_file()

            EXIT

        ELSE

            PRINT "Invalid selection. Please try again."

        ENDIF

    ENDDO

END OF FUNCTION
```

3.2.2 Apply leave function

```
DECLARE apply_leave()

    DECLARE leave_type, selection

    DOWHILE (selection is not e)

        SYSTEM clear screen

        CALL title()
```

PRINT menu elements

PRINT current leave balance

IF (staff[staff_select].balance less than or equal to 0)

 PRINT ‘No balance remaining.’

 BREAK

ELSE

 PRINT menu elements

 PRINT “Selection:”

 ACCEPT selection

 IF selection is 1 to 5

 DECLARE year, month, day, flag = 0

 DOWHILE (flag is 0)

 SYSTEM clear screen

 CALL title()

 PRINT “Year:”

 ACCEPT year

 PRINT “Month:”

 ACCEPT month

 PRINT “Day:”

 ACCEPT day

 IF (CALL dayValid(year, month, day))

 ASSIGN year, month, day, to structure

 staff[staff_select].status[inverted_balance] = 1

```
        SYSTEM clear screen

        CALL title()

        PRINT "Application successful"

        IF type is 1

            PRINT "Annual leave"

        ELSE IF type is 2

            PRINT "Compassionate leave"

        ELSE IF type is 3

            PRINT "Emergency leave"

        ELSE IF type is 4

            PRINT "Maternity leave"

        ELSE IF type is 5

            PRINT "Medical leave"

        ENDIF

        PRINT leave date

        PRINT leave status

        PRINT balance

    ELSE

        PRINT "Invalid."

    ENDIF

ENDDO

ELSE IF selection is b

    RETURN
```

```
        ELSE IF selection is e
            PRINT "Exiting..."
            CALL write_info_file()
            CALL write_leave_file()
            EXIT
        ELSE
            PRINT "Invalid selection. Please try again."
        ENDIF
    ENDDO
END OF FUNCTION
```

3.2.3 Day validity checker function

```
DECLARE dayValid(year, month, day)

    DECLARE month_length[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

    IF (year is current_year)

        IF (month is greater than current_month and month is less than 12)

            IF (day is greater than month_length[month])

                RETURN true

            ENDIF

        ELSE

            IF (month is current_month)

                IF (day less than or equal to month_length[month] and day is greater
                    than current day)
```

```

                                RETURN
                            ENDIF
                        ENDIF
                    ENDIF
                ENDIF
            IF (year is greater than current_year)
                IF (month less than or equal to 12)
                    IF (day less than or equal to month_length[month])
                        RETURN true
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    RETURN false
END OF FUNCTION
```

3.2.4 Cancel leave function

```

DECLARE cancel_leave()

    DECLARE selection, int_select, select_move

    DOWHILE (selection is not e)

        DECLARE flag = 1

        SYSTEM clear screen

        CALL title()

        PRINT header text
```

```
CALL leave_list()

PRINT menu items

PRINT "Selection:"

ACCEPT selection

int_select = selection - 1

select_move = selection

IF (select_move greater than 0 and less than or equal to inverted balance)

    IF (year less than current_year)

        PRINT "Date has already passed."

        flag = 0

    ELSE

        IF (month < current_month and year = current_year)

            PRINT "Date has already passed."

            flag = 0

        ELSE

            IF (day less than or equal to current_day and month is
            current_month and year is current_year)

                PRINT "Date has already passed."

                flag = 0

            ENDIF

        ENDIF

    ENDIF

ENDIF

IF (flag is 1)
```



```
        ASSIGN type, year, month, day, status to 0

        FOR (int_select to less than inverted_balance) STEP 1

            select_move = select_move + 1

            ASSIGN next type, year, month, day, status to current type,
                year, month, day, status

        ENDFOR

        inverted_balance = inverted_balance - 1

        balance = balance + 1

    ENDIF

ELSE IF selection is b

    RETURN

ELSE IF selection is e

    PRINT "Exiting..."

    CALL write_info_file()

    CALL write_leave_file()

    EXIT

ELSE

    PRINT "Invalid selection. Please try again."

ENDIF

ENDDO

END OF FUNCTION
```

3.2.5 Leave status function

```
DECLARE leave_status()  
  
    SYSTEM clear screen  
  
    CALL title()  
  
    PRINT menu items  
  
    CALL leave_list()  
  
    PRINT remaining balance  
  
END OF FUNCTION
```

3.2.6 Leave list function

```
DECLARE leave_list()  
  
    DECLARE i  
  
    PRINT header  
  
    FOR (i = 0 to inverted_balance) STEP 1  
        IF type is 1  
            PRINT "Annual leave"  
        ELSE IF type is 2  
            PRINT "Compassionate leave"  
        ELSE IF type is 3  
            PRINT "Emergency leave"  
        ELSE IF type is 4  
            PRINT "Maternity leave"  
        ELSE IF type is 5
```

```
        PRINT "Medical leave"

    ENDIF

    PRINT date

    IF status is 1

        PRINT "Pending"

    ELSE IF status is 2

        PRINT "Approved"

    ELSE IF status is 3

        PRINT "Rejected"

    ENDIF

ENDFOR

RETURN

END OF FUNCTION
```

3.3 Supervisor file

3.3.1 Supervisor menu function

```
DECLARE supervisor_menu()
```

```
    DECLARE selection
```

```
    DOWHILE (selection is not e)
```

```
        SYSTEM clear screen
```

```
        CALL title()
```

```
        PRINT menu items
```

```
        CALL user_info()
```

```
        PRINT menu options
```

```
        PRINT "Selection:"
```

```
        ACCEPT selection
```

```
        IF selection is 1
```

```
            CALL apply_leave()
```

```
        ELSE IF selection is 2
```

```
            CALL cancel_leave()
```

```
        ELSE IF selection is 3
```

```
            CALL leave_status()
```

```
        ELSE IF selection is 4
```

```
            CALL application_management()
```

```
        ELSE IF selection is 5
```

```
            CALL date_lookup()
```

```
        ELSE IF selection is b

            CALL main_menu()

        ELSE IF selection is e

            PRINT "Exiting..."

            CALL write_info_file()

            CALL write_leave_file()

            EXIT

        ELSE

            PRINT "Invalid selection. Please try again."

        ENDIF

    ENDDO

END OF FUNCTION
```

3.3.2 Application management function

```
DECLARE application_management()

    DECLARE selection, i, j, int_select

    DOWHILE (selection is not e)

        DECLARE count = 1

        SYSTEM clear screen

        CALL title()

        PRINT header text

        IF dept is 1

            PRINT "Administrative"
```

```
ELSE IF dept is 2
    PRINT "Management"
ELSE IF dept is 3
    PRINT "Academic"
ELSE IF dept is 4
    PRINT "Technical"
ENDIF

PRINT header texts

FOR (i = 0 to staff_count) STEP 1
    IF (staff[i].dept is staff[staff_select].dept)
        FOR (j = 0 to staff[i].inverted_balance) STEP 1
            IF (status is 1)
                PRINT count
                count = count + 1
                PRINT username, staff_id, day, month, year
            ENDIF
        ENDFOR
    ENDIF
ENDFOR

PRINT menu options

PRINT "Selection:"

ACCEPT selection

int_select = selection - 1
```

```
    IF (int_select more than or equal to 0 and less than count – 1)

        CALL application_processing(int_select)

        int_select = -1

    ELSE IF selection is b

        RETURN

    ELSE IF selection is e

        PRINT “Exiting...”

        CALL write_info_file()

        CALL write_leave_file()

        EXIT

    ELSE

        PRINT “Invalid selection. Please try again.”

    ENDIF

ENDDO

END OF FUNCTION
```

3.3.3 Application processing menu function

```
DECLARE application_processing(int_select)

    DECLARE i, j, count = 0, flag = 0

    DECLARE i_select = -1, j_select = -1

    DECLARE selection

    SYSTEM clear screen

    CALL title()
```

```
FOR (i = 0 to staff_count) STEP 1

    IF (staff[i].dept is dept)

        FOR (j = 0 to inverted_balance) STEP 1

            IF (staff[i].status[j] is 1)

                IF (count is int_select and flag is 0)

                    PRINT header text

                    PRINT username, staff_id, day, month, year, type

                    IF type is 1

                        PRINT "Annual leave"

                    ELSE IF type is 2

                        PRINT "Compassionate leave"

                    ELSE IF type is 3

                        PRINT "Emergency leave"

                    ELSE IF type is 4

                        PRINT "Maternity leave"

                    ELSE IF type is 5

                        PRINT "Medical leave"

                    ENDIF

                    i_select = i

                    j_select = j

                    flag = 1

                ENDIF

                count = count + 1
```



```
                ENDIF
            ENDFOR
        ENDIF
    ENDFOR
    PRINT menu items
    PRINT "Selection:"
    ACCEPT selection
    IF selection is 1
        staff[i_select].status[j_select] = 2
    ELSE IF selection is 2
        staff[i_select].status[j_select] = 3
    ELSE IF selection is b
        RETURN
    ELSE
        PRINT "Invalid selection."
    ENDIF
END OF FUNCTION
```

3.3.4 Date lookup function

DECLARE date_lookup()

 DECLARE year, month, day, i, j, count = 1

 SYSTEM clear screen

 CALL title()

 PRINT “Year:”

 ACCEPT year

 PRINT “Month:”

 ACCEPT month

 PRINT “Day:”

 ACCEPT day

 PRINT header text

 FOR (i = 0 to staff_count) STEP 1

 IF (staff[i].dept is dept)

 FOR (j = 0 to inverted_balance) STEP 1

 IF (year, month, and day matches)

 PRINT count

 PRINT username, staff_id

 IF type is 1

 PRINT “Annual leave”

 ELSE IF type is 2

 PRINT ‘Compassionate leave’

 ELSE IF type is 3

```
        PRINT "Emergency leave"

    ELSE IF type is 4

        PRINT "Maternity leave"

    ELSE IF type is 5

        PRINT "Medical leave"

    ENDIF

    IF status is 1

        PRINT "Pending"

    ELSE IF status is 2

        PRINT "Approved"

    ELSE IF status is 3

        PRINT "Rejected"

    ENDIF

    ENDIF

    ENDFOR

    ENDIF

    ENDFOR

    PRINT count

END OF FUNCTION
```

3.4 Admin file

3.4.1 Admin menu function

DECLARE admin_menu()

 DECLARE selection

 DOWHILE (selection is not e)

 SYSTEM clear screen

 CALL title()

 PRINT menu items

 CALL user_info()

 PRINT menu options

 PRINT "Selection:"

 ACCEPT selection

 IF selection is 1

 CALL apply_leave()

 ELSE IF selection is 2

 CALL cancel_leave()

 ELSE IF selection is 3

 CALL leave_status()

 ELSE IF selection is 4

 CALL add_staff()

 ELSE IF selection is 5

 CALL edit_balance()

 ELSE IF selection is 6

```
        CALL staff_information()

    ELSE IF selection is 7

        CALL generate_report()

    ELSE IF selection is b

        CALL main_menu()

    ELSE IF selection is e

        PRINT "Exiting..."

        CALL write_info_file()

        CALL write_leave_file()

        EXIT

    ELSE

        PRINT "Invalid selection. Please try again."

    ENDIF

ENDDO

END OF FUNCTION
```

3.4.2 Staff information lookup function

```
DECLARE staff_information()

    DECLARE info_lookup[max_staff_id_length]

    DECLARE i, lookup_select, flag = 0

    SYSTEM clear screen

    CALL title()

    PRINT header text

    PRINT "Staff ID:"

    ACCEPT info_lookup

    FOR (i = 0 to staff_count) STEP 1

        IF (staff[i].staff_id same as info_lookup)

            lookup_select = i

            flag = 1

            PRINT header text

            PRINT username, staff_id

            IF perm_level is 1

                PRINT "Staff"

            ELSE IF perm_level is 2

                PRINT "Supervisor"

            ELSE IF perm_level is 3

                PRINT "Administrator"

            ENDIF

        IF dept is 1
```

```
        PRINT "Administrative"

ELSE IF dept is 2

        PRINT "Management"

ELSE IF dept is 3

        PRINT "Academic"

ELSE IF dept is 4

        PRINT "Technical"

ENDIF

PRINT header files

FOR (i = 0 to staff[lookup_select].inverted_balance) STEP 1

    PRINT i + 1

    IF type is 1

        PRINT "Annual leave"

    ELSE IF type is 2

        PRINT "Compassionate leave"

    ELSE IF type is 3

        PRINT "Emergency leave"

    ELSE IF type is 4

        PRINT "Maternity leave"

    ELSE IF type is 5

        PRINT "Medical leave"

    ENDIF

PRINT date
```

```
        IF status is 1
            PRINT "Pending"
        ELSE IF status is 2
            PRINT "Approved"
        ELSE IF status is 3
            PRINT "Rejected"
        ENDIF
    ENDFOR
ENDIF
ENDFOR
IF (flag is 0)
    PRINT "Invalid selection, please try again."
ENDIF
END OF FUNCTION
```

3.4.3 Generate monthly report function

```
DECLARE generate_report()
    DECLARE selection, dept_char[10]
    DECLARE year, month, dept_int, i, j, count = 0
    DOWHILE (selection is not e)
        SYSTEM clear screen
        CALL title()
        PRINT menu text
```



```
PRINT "Selection:"

ACCEPT selection

IF selection is y

    PRINT "Year:"

    ACCEPT year

    PRINT "Month:"

    ACCEPT month

    PRINT dept list

    ACCEPT dept_int

    IF dept_int is 1

        dept_char = admin

    ELSE IF dept_int is 2

        dept_char = manage

    ELSE IF dept_int is 3

        dept_int = acad

    ELSE IF dept_int is 4

        dept_int = tech

ENDIF

DECLARE report_name[50]

CREATE monthly report file

IF (report is NULL)

    PRINT "Failed to generate text file!"

ENDIF
```

```
OPEN file as report

FOR (i = 0 to staff_count) STEP 1

    IF (staff[i].dept same as dept_int)

        FOR (j = 0 to staff[i].inverted_balance) STEP 1

            count = count + 1

            WRITE username, staff_id into file

            WRITE perm_level into file

            WRITE type into file

            WRITE leave date into file

            WRITE status into file

        ENDFOR

    ENDIF

ENDFOR

PRINT "File generated!"

ELSE IF selection is b

    CALL main_menu()

ELSE IF selection is e

    PRINT "Exiting..."

    CALL write_info_file()

    CALL write_leave_file()

    EXIT

ELSE

    PRINT "Invalid selection. Please try again."
```

ENDIF

ENDDO

END OF FUNCTION

3.4.4 Edit leave balance function

DECLARE edit_balance()

DECLARE selection, edit_day, confirmation[5]

DECLARE i, new_leave

DOWHILE (selection is no e)

SYSTEM clear screen

CALL title()

PRINT menu items

PRINT "Selection:"

ACCEPT selection

IF selection is 1

SYSTEM clear screen

CALL title()

PRINT menu items

PRINT "Selection:"

ACCEPT edit_day

IF edit_day is 1

FOR (i = 0 to staff_count) STEP 1

staff[i].balance = staff[i].balance + 1

```
        ENDFOR

        PRINT "Added 1 day."

    ELSE IF edit_day is 2

        FOR (i = 0 to staff_count) STEP 1

            staff[i].balance = staff[i].balance - 1

        ENDFOR

        PRINT "Removed 1 day."

    ELSE IF edit_day is b

        BREAK

    ELSE

        PRINT "Invalid selection."

    ENDIF

ELSE IF selection is 2

    SYSTEM clear screen

    CALL title()

    PRINT header texts

    PRINT "Confirmation:"

    ACCEPT confirmation

    IF (confirmation is YES)

        PRINT header text

        PRINT "Leave days:"

        ACCEPT new_leave

        FOR (i = 0 to staff_count) STEP 1
```

```
        staff[i].balance = new_leave

    ENDFOR

    PRINT "Balances updated."

ELSE

    PRINT "Action cancelled."

ENDIF

ELSE IF selection is b

    RETURN

ELSE IF selection is e

    PRINT "Exiting..."

    CALL write_info_file()

    CALL write_leave_file()

    EXIT

ELSE

    PRINT "Invalid selection. Please try again."

ENDIF

ENDDO

END OF FUNCTION
```

3.4.5 Add staff information function

DECLARE add_staff()

 DECLARE selection, confirmation[5], dept_selection, rank_selection

 DOWHILE (selection is no e)

 SYSTEM clear screen

 CALL title()

 PRINT header text

 PRINT "Confirmation:"

 ACCEPT confirmation

 IF (confirmation is YES)

 SYSTEM clear screen

 CALL title()

 PRINT header text

 PRINT "Username:"

 ACCEPT username

 PRINT "Staff ID:"

 ACCEPT staff_id

 PRINT "Password:"

 ACCEPT password

 PRINT department list

 PRINT "Selection:"

 ACCEPT dept_selection

 IF dept_selecton is 1

```
        staff[staff_count].dept = 1

ELSE IF dept_selection is 2

        staff[staff_count].dept = 2

ELSE IF dept_selection is 3

        staff[staff_count].dept = 3

ELSE IF dept_selection is 4

        staff[staff_count].dept = 4

ELSE

        PRINT "Invalid selection."

        RETURN

ENDIF

PRINT staff rank list

PRINT "Selection:"

ACCEPT rank_selection

IF rank_selection is 1

        staff[staff_count].perm_level = 1

ELSE IF rank_selection is 2

        staff[staff_count].perm_level = 2

ELSE IF rank_selection is 3

        staff[staff_count].perm_level = 3

ELSE

        PRINT "Invalid selection."

        RETURN
```

```
ENDIF

PRINT "Leave balance:"

ACCEPT leave_balance

SYSTEM clear screen

CALL title()

PRINT "Staff added."

PRINT new staff information

ELSE

PRINT "Action cancelled."

ENDIF

ENDDO

END OF FUNCTION
```


4.0 Program Source Code Concepts and Explanations

Throughout this assignment, various C programming concepts were utilized to implement the functions offered by the program. The following section provides examples of some of these concepts, as well as explanation on how they function and how were they implemented.

4.1 File Handling

The preprogrammed or user input data while running the program is stored on the RAM, which is volatile. Meaning that all the values that are not in the source code such as variables states and user inputs are lost when the program is closed or if the computer suddenly loses power and is not recoverable on the following session. To address this issue, the data has to be stored on a non-volatile storage medium which is the computer's hard drives or solid state drives. In this program, this is accomplished by storing data on text, or .txt files.

Since the files are not stored locally in the program source code itself, all the data must be imported from the text files. First, the selected text files are opened, and the program has to interpret and sort the data in the text file into data structures that can be understood and used by the program. To avoid potential complications, it is best that the text file be closed after importing is completed.

Upon completion of a session on the program, the newly modified data must be stored back into the text files to preserve persistence into the next session. To accomplish this, the text file must be opened, and the data on the data structures must be neatly printed into the text file in a format that can be interpreted by the program on next startup. The text files are then reclosed prior to the exiting of the program.

The following contain examples on how this program handles data and text files:

```

//Read staff info from text file
void read_info_file() {
    int i;
    FILE* staffinfo;
    fopen_s(&staffinfo, "staff_info.txt", "r");
    if (staffinfo == NULL) {
        red_bold(); printf("\n\tFailed to read from file!"); clear();
        _getch();
    }
    for (i = 0; !feof(staffinfo); i++) {
        fscanf_s(staffinfo, "%s %s %s %d %d %d %d", staff[i].username, sizeof(staff[i].username), staff[i].staff_id, sizeof(staff[i].staff_id),
        staff[i].password, sizeof(staff[i].password), &staff[i].perm_level, &staff[i].balance, &staff[i].invbalance, &staff[i].dept);
    }
    STAFF_COUNT = i - 1;
    fclose(staffinfo);
}

```

Figure 4.1.1: *Read staff info from text file*

The above figure 4.1.1 contains an extract from the source code of the program. It is a function that reads the staff information from the text files and sorts them into the program's struct data structure to be used down the line. This function is executed upon startup of the program.

First, opens a file named "staff_info.txt" as "staffinfo" in the read format. Then, it checks for whether the previous function has been successfully performed. In case the text file is empty, an error is displayed.

Else, the program function then reads through the text file line by line while scanning the contents of the line into their respective slots in the struct staff data structure. Each line in the text file represents one user data and the various elements of a user data are separated with a space. This step is repeated until the end of the file is reached by using a for loop. This function also interprets the staff count by reading the number of loops the for loop cycled through to reach the end of the file.

```

default default default123 3 19 1 1
supervisor supervisor1 supervisor123 2 20 0 1
admin admin1 admin123 3 20 0 1
admin admin2 admin123 3 20 0 1
admin admin3 admin123 3 20 0 1
supervisor supervisor2 supervisor123 2 20 0 2
staff staff2 staff123 1 15 6 2
staff staff3 staff123 1 19 1 2
supervisor supervisor3 supervisor123 2 20 0 3
staff staff4 staff123 1 20 0 3
staff staff5 staff123 1 20 0 3
supervisor supervisor3 supervisor123 2 20 0 3
staff staff6 staff123 1 20 0 4
test test1 test123 1 20 0 2

```

Figure 4.1.2: *Staff info text file*

The above figure 4.1.2 is the contents of the user info text file. Due to system limitations, there must be at least one user in the program for the program to run. Hence, the default user at the very top. The default user also acts as a backup account in case all other accounts were erased.

Going by order from left to right, the data in a single line of the text file are as follows: username, staff ID, user password, user rank in integer, user leave balance, inverted user leave balance, user department.

As the text files are not meant for human readability, the user rank and department are stored in integer form for easy interpretation in the program. For user rank 1, the user is of the staff rank, 2 for supervisor, and 3 for admin. For the department, 1 is administrative, 2 is management, 3 is academic, and 4 is technical.

```
//Write staff info into text file
void write_info_file() {
    FILE* staffinfo;
    fopen_s(&staffinfo, "staff_info.txt", "w");
    if (staffinfo == NULL) {
        printf("Failed to open file for writing.\n");
        return;
    }
    for (int i = 0; i < STAFF_COUNT; i++) {
        fprintf(staffinfo, "%s %s %s %d %d %d %d\n",
            staff[i].username, staff[i].staff_id, staff[i].password,
            staff[i].perm_level, staff[i].balance, staff[i].invbalance, staff[i].dept);
    }
    fclose(staffinfo);
}
```

Figure 4.1.3: *Write staff info into text file*

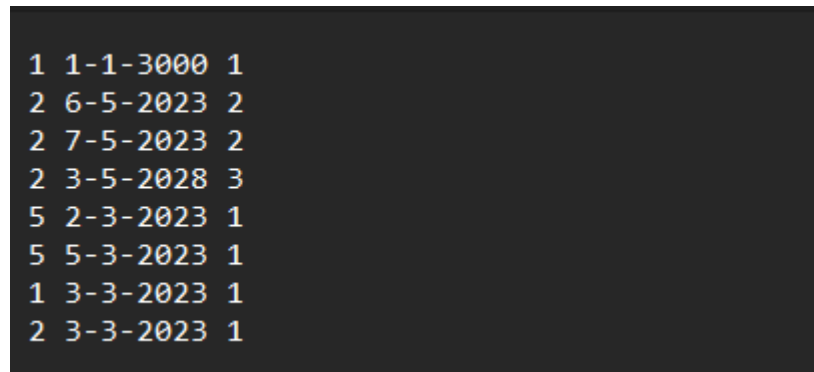
The above figure 4.1.3 is the write staff info into text file function. As the name suggests, this program writes all the data in the staff struct data structure into the text file for permanent storage. This function is executed when the user selects an “Exit” option from the user menus.

This function writes all the staff information into the text file in the exact same format as shown above to ensure readability by the program on next startup using space as separators for elements in a user’s info and lines as separators for different users’ data.

A total of 2 text files used in the program, one for staff information, and one for leave information. Each with 2 functions to scan and print data in and out of them.

4.2 Leave information file data sorting

The sorting format of the text file for leave information is a little different and unique. Instead of using each line as an indicator for a new user data, it instead reads the file based on the inverted user leave balance data retrieved from the staff info text file.



```

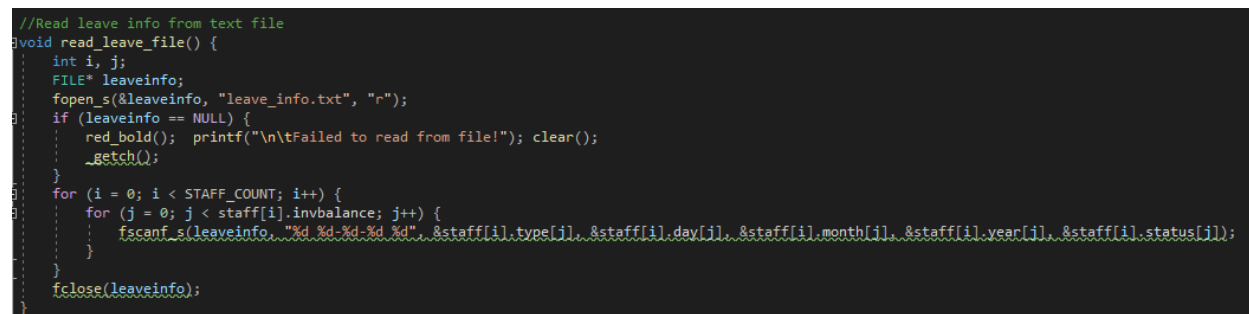
1 1-1-3000 1
2 6-5-2023 2
2 7-5-2023 2
2 3-5-2028 3
5 2-3-2023 1
5 5-3-2023 1
1 3-3-2023 1
2 3-3-2023 1

```

Figure 4.2.1: *Leave info file contents*

The above figure 4.2.1 showcases the contents of the leave info text file. Each line represents one set of leave information. The first element from the left is the leave type, followed by the date of the leave in DD-MM-YYYY format, then lastly is the leave status. The leave info file contains no information as to which line of data belongs to which user as the sorting is handled elsewhere by the program.

As the name suggests, the inverted leave balance is the inverse of the user's leave balance. For example: The default leave balance is 20, so the leave balance is 20 and the invert balance is 0. If the user applied for one day of leave, the new balance is 19 and the invert balance is 1. This variable is used to determine how many lines of leave data in the leave info file belongs to a certain user.



```

//Read leave info from text file
void read_leave_file() {
    int i, j;
    FILE* leaveinfo;
    fopen_s(&leaveinfo, "leave_info.txt", "r");
    if (leaveinfo == NULL) {
        red_bold(); printf("\n\tFailed to read from file!"); clear();
        _getch();
    }
    for (i = 0; i < STAFF_COUNT; i++) {
        for (j = 0; j < staff[i].invbalance; j++) {
            fscanf_s(leaveinfo, "%d %d-%d-%d %d", &staff[i].type[j], &staff[i].day[j], &staff[i].month[j], &staff[i].year[j], &staff[i].status[j]);
        }
    }
    fclose(leaveinfo);
}

```

Figure 4.2.2: *Read from leave info text file function*

The above figure 4.2.2 contains the read from leave info text file function. After opening and checking the validity of the text file, the program then loops through each of the staff member's info in the staff struct and reads the inverted leave balance of each of the user.

For example: If the inverted leave balance is 0, it means that none of the leave info in the text file belongs to that user. Then, it moves on to the next user and does the same. If the inverted balance is 1, then the 1st line of data in the leave info file is read and assigned to that user. If the next user has an inverted balance of 3, then data in lines 2 to 4 of the leave info text file belongs to the leave list under the user's name in the staff struct. This process is repeated until all the staff members have been cycled through and the leave info text file is exhausted.

```
//Write leave info into text file
void write_leave_file() {
    int i, j;
    FILE* leaveinfo;
    fopen_s(&leaveinfo, "leave_info.txt", "w");
    if (leaveinfo == NULL) {
        red_bold(); printf("\n\tFailed to read from file!"); clear();
        _getch();
    }
    for (i = 0; i < STAFF_COUNT; i++) {
        for (j = 0; j < staff[i].invbalance; j++) {
            fprintf(leaveinfo, "%d-%d-%d-%d\n", staff[i].type[j], staff[i].day[j], staff[i].month[j], staff[i].year[j], staff[i].status[j]);
        }
    }
    fclose(leaveinfo);
}
```

Figure 4.2.3: *Write leave info into text file*

The above figure 4.2.3 contains the function responsible for writing the leave info from the staff struct into the text file to preserve persistence. Similar to how the program reads the leave info from the text file, a for loop is used to cycle through all of the staff members and their list of leave applications to print into the text file.

4.3 Converting character data type into integer data type

```
scanf_s("%c", &selection); while (getchar() != '\n');  
  
//Assign character selection to integer variables  
sscanf_s(&selection, "%d", &select_move);
```

Figure 4.3.1: *Converting character to integer*

Throughout the program, all user inputs are accepted in character form. The decision to use character instead of integer was made to allow for user inputs of ‘b’ for back and ‘e’ for exit. While an integer could be assigned for these functions as well, it was decided that having universal keys for back and exit commands that remains unchanged regardless of the menu is more important as it significantly increased the ease of menu navigation.

However, sometimes the user inputs has to be in integer form for processing and value comparisons. Thus, a method was used to assign the user input to another variable with an integer data type where applicable. To accomplish this, the `sscanf_s` function was used which reads the data from a buffer location and assign the value to a new location with a specified data format. From the example in figure 4.3.1, the function is provided with a pointer to the variable `selection`’s memory location and told to assign its contents into the memory location of the `select_move` variable with data type integer.

With this conversion, if the user inputs an integer, the integer can be used in processing such as running for loops while allowing users to input characters for other functions in the same input field.

4.4 Function with arguments

```
//Day validity checker
bool dayValid(int year, int month, int day) {
    int month_len[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    if (year == current_year) {
        if (month > current_month && month <= 12) {
            if (day <= month_len[month]) {
                return true;
            }
        }
        else {
            if (month == current_month) {
                if (day <= month_len[month] && day > current_day) {
                    return true;
                }
            }
        }
    }
    if (year > current_year) {
        if (month <= 12) {
            if (day <= month_len[month]) {
                return true;
            }
        }
    }
    return false;
}
```

Figure 4.4.1: *Function definition of the day validity checker with arguments*

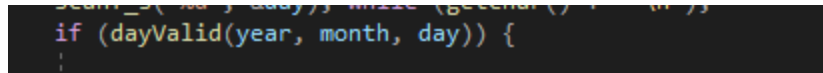
Function arguments are variables or user inputs that are passed to a function to provide information for the function to operate on. The above figure 4.4.1 is a function definition with 3 arguments that returns either a true or false. The function is used to check the validity of the user input date during leave application as it checks for whether the selected date has already passed.

In this function, the 3 arguments used are the year, month, and day, all with the data type of integer and are passed down to the function by value.

Figure 4.4.2 below is the function prototype of the dayValid function with the argument data types specified as data type integer.

```
bool dayValid(int, int, int);
```

Figure 4.4.2: *Function prototype*



```

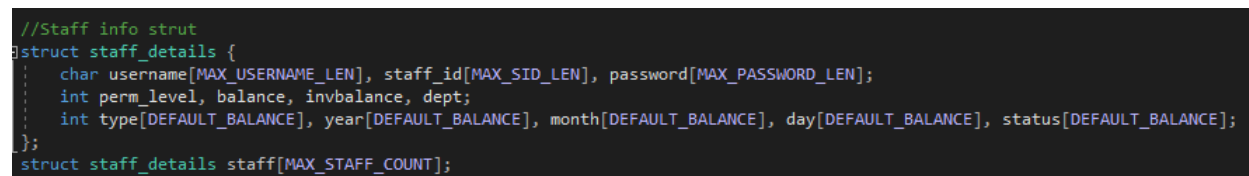
if (dayValid(year, month, day)) {

```

Figure 4.4.3: *Function call*

The above figure is the function call of the dayValid function where the 3 arguments are specified as the variables year, month, and day of which the user had inputted prior to the function call. The function call resides in an if statement due to the function having a boolean return type.

4.5 C structs



```

//Staff info struct
struct staff_details {
    char username[MAX_USERNAME_LEN], staff_id[MAX_SID_LEN], password[MAX_PASSWORD_LEN];
    int perm_level, balance, invbalance, dept;
    int type[DEFAULT_BALANCE], year[DEFAULT_BALANCE], month[DEFAULT_BALANCE], day[DEFAULT_BALANCE], status[DEFAULT_BALANCE];
};
struct staff_details staff[MAX_STAFF_COUNT];

```

Figure 4.5.1: *C struct used in the program*

There are many data structures that can be used in the C language. In this program, structs with embedded arrays for the leave information are used. Structs are a way of grouping together multiple variables with different data types under a common name. The variables grouped under a struct are called members.

The above figure 4.5.1 showcases the struct definition for the staff_details struct which is used throughout the program. The first line declares the character arrays, or strings, in the struct which is the username, staff ID, and the user password. Next, it declares the members for the staff's rank, leave balance, inverted leave balance, and department all in integer form as the decoding of the integers is handled elsewhere in the program. Lastly, the leave information for each of the staff are declared as integer arrays.

The last line in figure 4.5.1 is the struct declaration which declares the staff_details struct as "staff" to be used throughout the program. The usage of structs in this program allows all the data to be neatly organized and easily accessible by all the functions simply by referencing the staff struct with a dot operator such as the following: staff.status[]

4.6 ANSI colour codes

```
void clear() {  
    printf("\033[m");  
}  
  
void white() {  
    printf("\033[0;37m");  
}  
void white_bold() {  
    printf("\033[1;37m");  
}  
  
void red() {  
    printf("\033[0;31m");  
}  
void red_bold() {  
    printf("\033[1;31m");  
}
```

Figure 4.6.1: *Colour code function definitions*

Even though aesthetics of the user interface are not part of the assignment specifications, this program still includes some basic colors in the user interface to improve the menu readability. This is accomplished using the ANSI colour codes and are applied to the print commands through functions as shown in figure 4.6.1 above.

```
yellow_bold(); printf("\n\n\t\t\tMain Menu"); clear();
```

Figure 4.6.2: *Example of colour usage*

To apply colours to the text, the function with the colour is called before the print command. The function will apply the colour to all subsequent text printed in the console. Thus, after the print command is executed, the colour has to be cleared by calling the clear function. An example is as shown in figure 4.6.2 above.

5.0 Sample Input/Output with Explanations

5.1 Starting menu and login

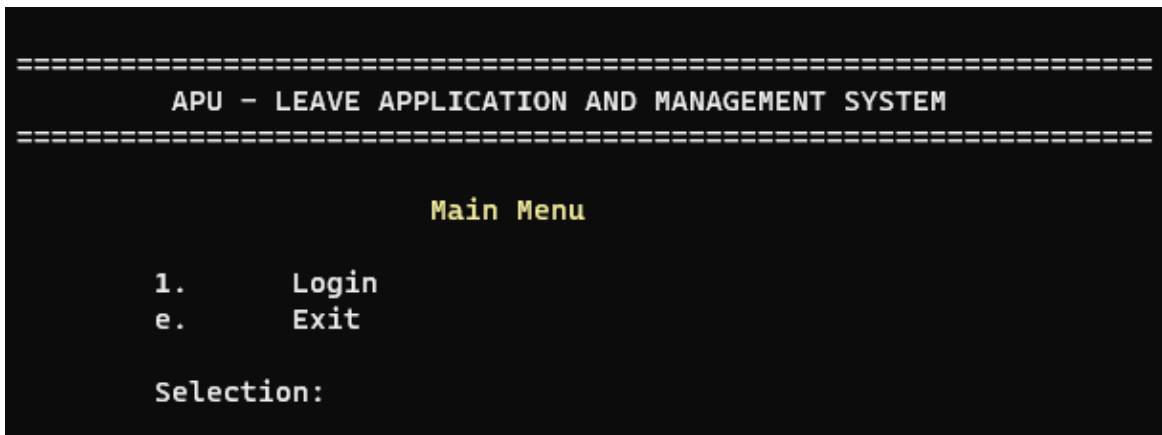


Figure 5.1.1: *Main menu*

The above figure 5.1.1 is the starting menu of this program. The user is greeted with the menu upon initializing the program. From the menu, the user may choose to proceed to login to their accounts or exit the program. Selection is done by inputting the corresponding numbers of characters on the left side of the choice text into the selection box below.

This menu, along with all other menus in this program where user inputs are required is fully equipped with error handling. In the event that the user inputs an integer, character, or string outside the expectations of the program, the program will give the user an error and asked the user to try again. This safety feature is demonstrated in figure 5.1.2 below.

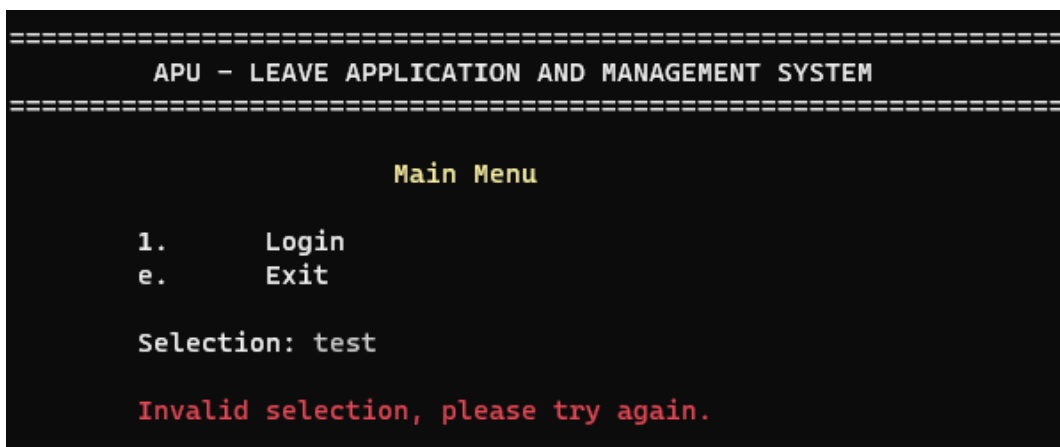


Figure 5.1.2: *Error handling*

Upon selecting the login function from the main menu, the user will be redirected to the login menu. Here, the user is prompted to input their staff ID and password as shown in figure 5.1.3 below.

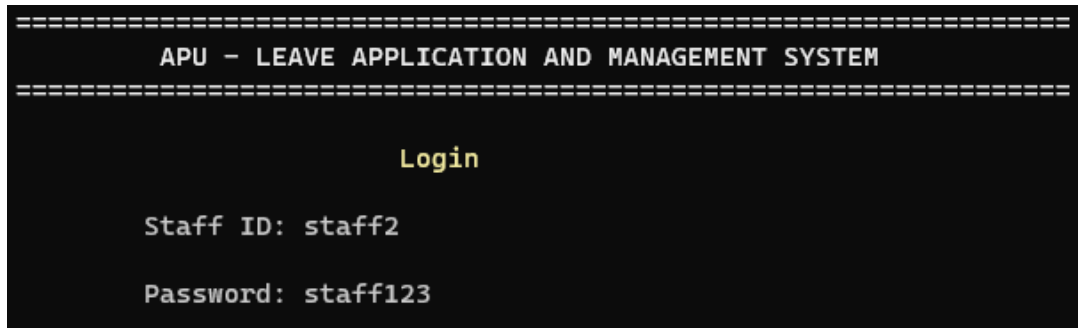


Figure 5.1.3: *Sample of staff ID and password*

The system will only grant entry to the user if the staff ID matches its corresponding password whereby afterwards, the system will automatically detect the rank of the user and redirects them accordingly.

If one or both inputs do not match with any records of the system, an error will be presented, and the user is allowed to retry as shown in figure 5.1.4 below.

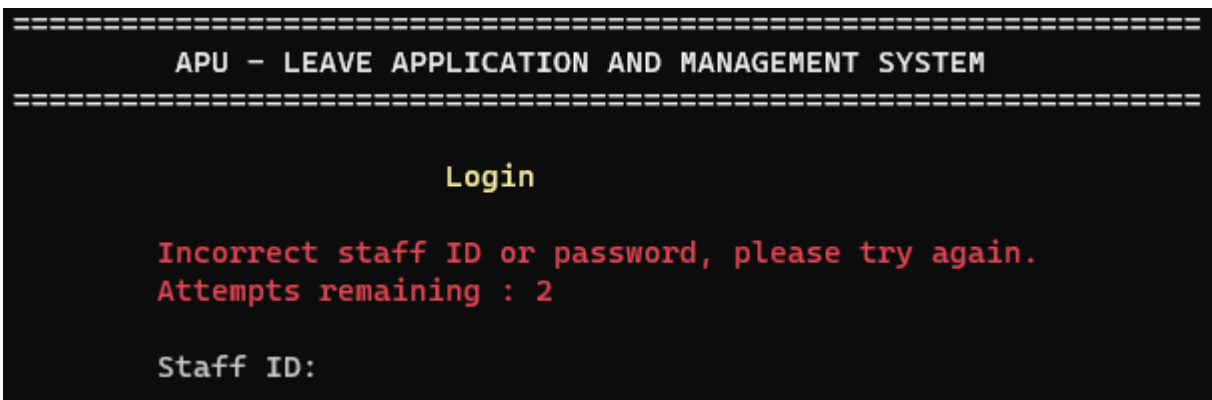


Figure 5.1.4: *Incorrect staff ID or password*

By default, the system allows a maximum of 3 attempts for logging in. If on the third attempt the user still inputs incorrect login credentials, the system will not allow any more attempts and will exit as shown in figure 5.1.5 below.

```
=====
      APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      Login

      Incorrect staff ID or password, please try again.
      Attempts remaining : 1

      Staff ID: test

      Password: test

      Incorrect staff ID or password, no attempts left.
      Exiting program...
```

Figure 5.1.5: *All attempts used*

5.2 Staff role

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      --- Staff Role ---

      User Info
Username:  staff
Staff ID:  staff2
User rank: Staff
Department Management
Leave balance: 20

      Leave Menu

1.      Apply leave
2.      Cancel leave
3.      Leave status and information

b.      Back
e.      Exit

Selection:
```

Figure 5.2.1: *Staff menu*

Upon a successful login, if the user is of the staff rank, they will be automatically redirected to this menu. This menu displays the user's basic information such as the username, staff ID, rank, department, and their remaining leave balance. The purpose of this display is to reaffirms the user that the system has correctly directed them to their accounts.

Below the user information display is the leave menu. This menu allows the user to apply for leave, cancel existing leave, as well as view their current leave statuses and information. As with most other menus in this program, the menu also has an option for going back a page and exiting the program.

To make a selection, the user inputs the integer or character of the respective menu items . For example, if the user wishes to apply for leave, the user should input '1' into the selection section under the menu.

5.3 Leave application menu

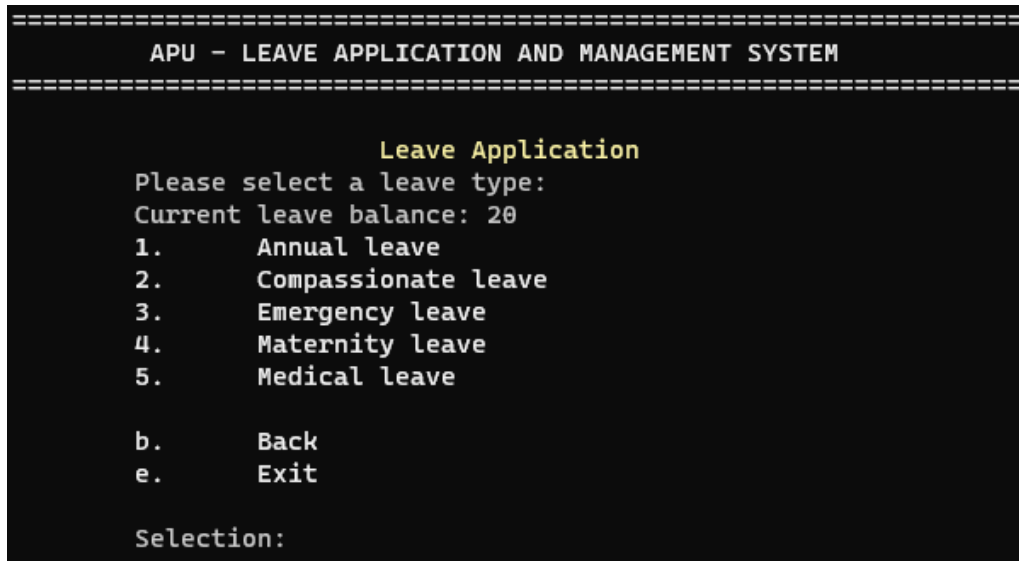


Figure 5.3.1: *Leave application menu*

The above figure 5.3.1 showcases the leave application menu. The user is redirected to this page upon selecting the leave application option in their user menu. This page is also accessible to members of the supervisor and administrator ranks from their respective menus which will be showcased further down the line.

This menu prompts the user to choose the nature of their leave which is broken down into 5 categories: annual, compassionate, emergency, maternity, and medical. As mentioned in the assumptions section above, all categories share the same leave balance pool. This menu also contains the last chance where the user can abort their leave application. Beyond this point, the user cannot go back while inputting their leave application details and will have to cancel leave through the cancel leave function.

Similar to all other menus in this program where user input is required, the user makes a selection simply by inputting the index integer or character in the selection box below. Upon making a selection that is not the back or exit option, the user will be directed to the next page where the user is prompted to input the year, month, and day of the leave as shown in figure 5.3.2 below.

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      Leave Date
Please enter the date of your leave.

Year: 2023

Month: 3

Day: 5
```

Figure 5.3.2: *Leave date input*

The program only allows users to apply leave for dates after the current date. If the user inputs the current date or a date that has already passed, the user will be given an error and then be prompted to try again as shown in figure 5.3.3 below:

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      Leave Date
Please enter the date of your leave.

Year: 2022

Month: 5

Day: 6

Invalid date, please try again.
```

Figure 5.3.3: *Invalid date*

Upon entering a valid date, the application is successful. Then, the user will be shown this menu where it displays all the leave information, along with the user's remaining leave balance. Pressing enter on this menu returns the user to the leave application menu where another application can be made or return to the staff menu.

```
=====
      APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Application successful!

Leave type:      Annual leave
Leave date:      5-3-2023
Leave status:    Pending approval
Balance:        19 days

Press enter to continue.
```

Figure 5.3.4: *Successful application*

5.4 Leave cancellation menu

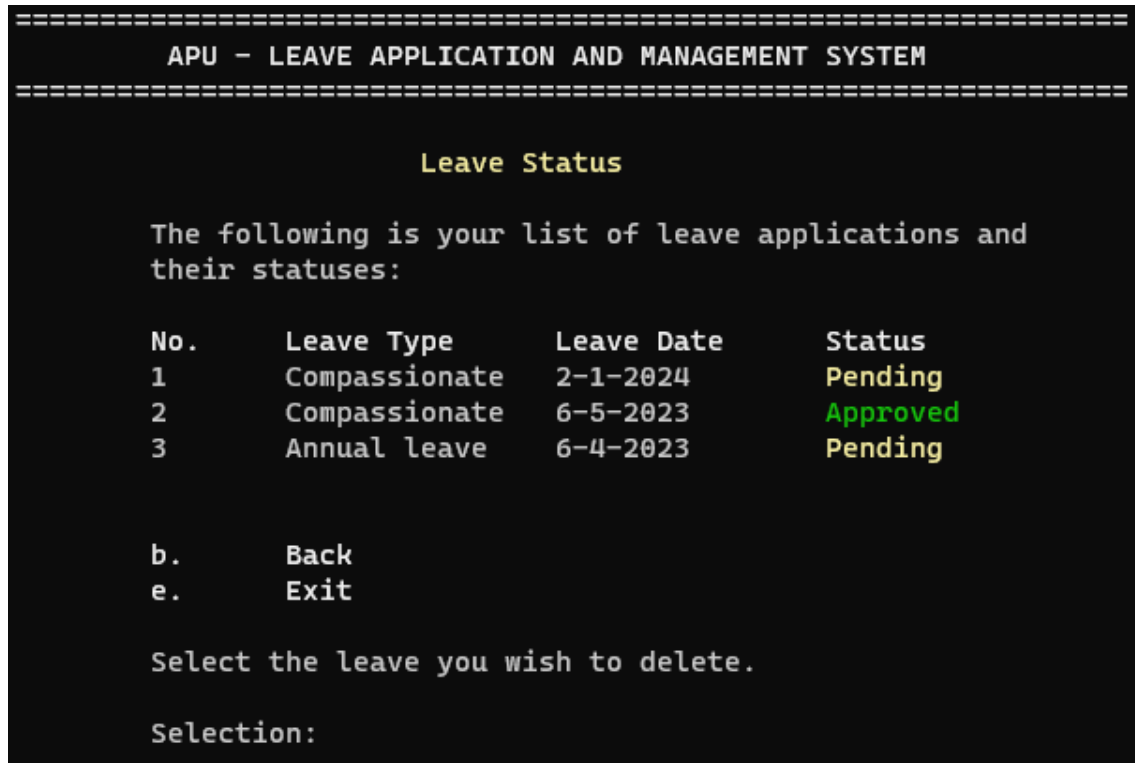


Figure 5.4.1: *Leave cancellation menu*

The above figure 5.4.1 shows the leave application menu. This menu is accessible by selecting the “Cancel leave” option from the user menu. At the top, it lists down all of the user’s leave applications along with its information such as the leave type, date, and status. The menu also has options to go back a menu or exit the program.

To cancel a leave, the user simply has to input the leave index number into the selection box below. Upon entering the leave number, the corresponding leave will immediately be removed provided that it is valid for removal. The only condition that a leave will not be removed is if the leave date has already passed.

For example, the user inputs the integer ‘1’ into the selection box to remove the compassionate leave applied for the 2nd of January, 2024. As the condition is still valid as of the time of writing, the leave will be removed as soon as the user hits enter as shown in figure 5.4.2 below.

```

=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Leave Status

The following is your list of leave applications and
their statuses:

No.      Leave Type      Leave Date      Status
1        Compassionate   6-5-2023        Approved
2        Annual leave    6-4-2023        Pending

b.       Back
e.       Exit

Select the leave you wish to delete.

Selection:

```

Figure 5.4.2: *Leave removed*

If a user inputs an index number not on the list, a warning will pop up informing the user of the error and prompts the user to try again. If the date of the leave the user is trying to cancel has already passed, it will prevent the user from doing so. Both outputs are shown in figures 5.4.3 and 5.4.4 respectively.

```

=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Leave Status

The following is your list of leave applications and
their statuses:

No.      Leave Type      Leave Date      Status
1        Compassionate   6-5-2023        Approved
2        Annual leave    6-4-2023        Pending

b.       Back
e.       Exit

Select the leave you wish to delete.

Selection: 3

Invalid selection, please try again.

```

Figure 5.4.3: *Invalid index*

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Leave Status

The following is your list of leave applications and
their statuses:

No.      Leave Type      Leave Date      Status
1        Compassionate  6-5-2023       Approved
2        Annual leave   6-4-2023       Pending
3        Compassionate  5-2-2023       Approved

b.       Back
e.       Exit

Select the leave you wish to delete.

Selection: 3

Date has already passed.
```

Figure 5.4.4: *Date of selection has already passed*

Upon a successful cancellation, a leave day will be refunded into the user's leave balance. However, if the leave application has already been rejected prior to cancellation, the user's leave balance will not change as the leave day has already been refunded to the user when the supervisor rejected the application.

5.5 Leave status and information menu

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      Leave Status and Information

The following is your list of leave applications and
their statuses:

No.      Leave Type      Leave Date      Status
1        Compassionate   6-5-2023       Approved
2        Compassionate   7-5-2023       Pending
3        Compassionate   3-5-2028       Rejected
4        Medical leave    2-3-2023       Pending
5        Medical leave    5-3-2023       Pending

Remaining leave balance:      16 days

Press enter to return to previous menu.|
```

Figure 5.5.1: *Leave status and information menu*

The above figure is the leave status and information menu. This menu can be accessed from the user menus and it displays all of the basic information of the user's leave days such as the leave type, date, status, as well as the remaining leave balance. This menu is only intended for viewing purposes and has no user interactable functions. Pressing enter will return the user to the previous menu.

5.6 Supervisor role

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      --- Supervisor Role ---

      User Info
Username:  supervisor
Staff ID:  supervisor2
User rank: Supervisor
Department Management
Leave balance: 20

      Leave Menu
1.      Apply leave
2.      Cancel leave
3.      Leave status and information

      Supervisor Menu
4.      Application management
5.      Date lookup

b.      Back
e.      Exit

Selection:
```

Figure 5.6.1: *Supervisor menu*

The above figure 5.6.1 displays the menu of users with the supervisor rank. Users with the supervisors rank are automatically directed to this menu upon their successful login. At the top, the user's information are displayed to give the user a reaffirmation that they have signed into the right account. Just below that is the leave menu which is the same as on the staff menu as well as the admin menu. Here, the user can apply for, cancel, or view their leave information.

The bottom menu contains the supervisor role exclusive functions which is the application management function and the date lookup function. This menu also contains the options to go back to the main menu as well as exit out of the program.

5.7 Application management menu

```

=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Application Management
Department Management

The following is the list of pending leave applications
for your department:

No.      Username      Staff ID      Leave Date
1.       staff        staff2        7-5-2023
2.       staff        staff2        2-3-2023
3.       staff        staff2        5-3-2023

b.       Back
e.       Exit

Select the application you wish to process.

Selection:

```

Figure 5.7.1: *Application management menu*

Upon selecting the application management option in the supervisor menu, the user is directed to this menu. Here, the user can view, approve, or reject leave applications of staff members in their department. The supervisor's department is shown at the top of the menu and the list of pending staff leave applications just below that.

The supervisor makes a selection by inputting the index number into the selection box and hitting enter. Then, it will bring the user to the menu in figure 5.7.2 where the supervisor can choose to approve or reject the application.

```

=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Username      Staff ID      Leave Date      Type
staff         staff2        7-5-2023        Compassionate

Choose whether you wish to approve or reject the application.

1.      Approve
2.      Reject

b.      Back

Selection: |

```

Figure 5.7.2: *Leave selected, awaiting decision*

Once a decision was made on the leave application, it will be automatically removed from the application management menu listing.

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Department      Application Management
                  Management

The following is the list of pending leave applications
for your department:

No.   Username   Staff ID   Leave Date
1.    staff     staff2     2-3-2023
2.    staff     staff2     5-3-2023

b.    Back
e.    Exit

Select the application you wish to process.

Selection: 3

Invalid selection, please try again.
```

Figure 5.7.3: *Invalid selection error handling*

As usual, the application management is also equipped with error handling for if the user inputs an invalid index number as shown in figure 5.7.3.

5.8 Date lookup menu

In addition to leave application management, users with the supervisor rank can also lookup a date and view all staff members in their departments who has applied for leave on that day. This feature can be accessed by selecting the “date lookup” function in the supervisor menu. Upon selection, the user will be directed to the menu as shown in figure 5.8.1 below. Here, the user is prompted into input a date that they wish to view the leave information of.

```

=====
      APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

                Date Lookup

Please enter a date you wish to view.

Year: 2023

Month: 3

Day: 3

```

Figure 5.8.1: *Date lookup input*

Once a date is entered, the system will display a list of staff members under the user’s department who has applied for leave on the selected date as well as the details of the leave as shown in figure 5.8.2.

```

=====
      APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

                Leave List

The following is the list of staff members on leave on the
selected date for your department.

No.      Username      Staff ID      Type      Status
1.       staff         staff2        Annual leave  Pending
2.       staff         staff3        Compassionate Pending

Number of staff on leave: 2

Press enter to return to previous menu.

```

Figure 5.8.2: *Leave list*

5.9 Admin role

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      --- Admin Role ---

      User Info
Username:  admin
Staff ID:  admin1
User rank: Administrator
Department: Administrative
Leave balance: 20

      Leave Menu
1.      Apply leave
2.      Cancel leave
3.      Leave status and information

      Admin Menu
4.      Add staff
5.      Edit leave balance
6.      Staff information
7.      Generate monthly report

b.      Back
e.      Exit

Selection:
```

Figure 5.9.1: *Admin menu*

The above figure showcases the user menu for users with the admin rank. Similar to previous ranks' user menus, the top is reserved for displaying the all the important user information. Just below that is the admin's personal leave application dashboard where they can apply, cancel, and check their leave statuses just like all other staff members.

At the bottom is the admin exclusive functions menu. From here, users with the admin rank can add new staff members, and edit the current leave balance for all staff which includes the manual function for entering a new year. In addition, admins can also lookup the information of a specific staff member by inputting their staff ID in the staff information menu, as well as generate a monthly report of all the staff on leave in the specified month.

5.10 Add staff

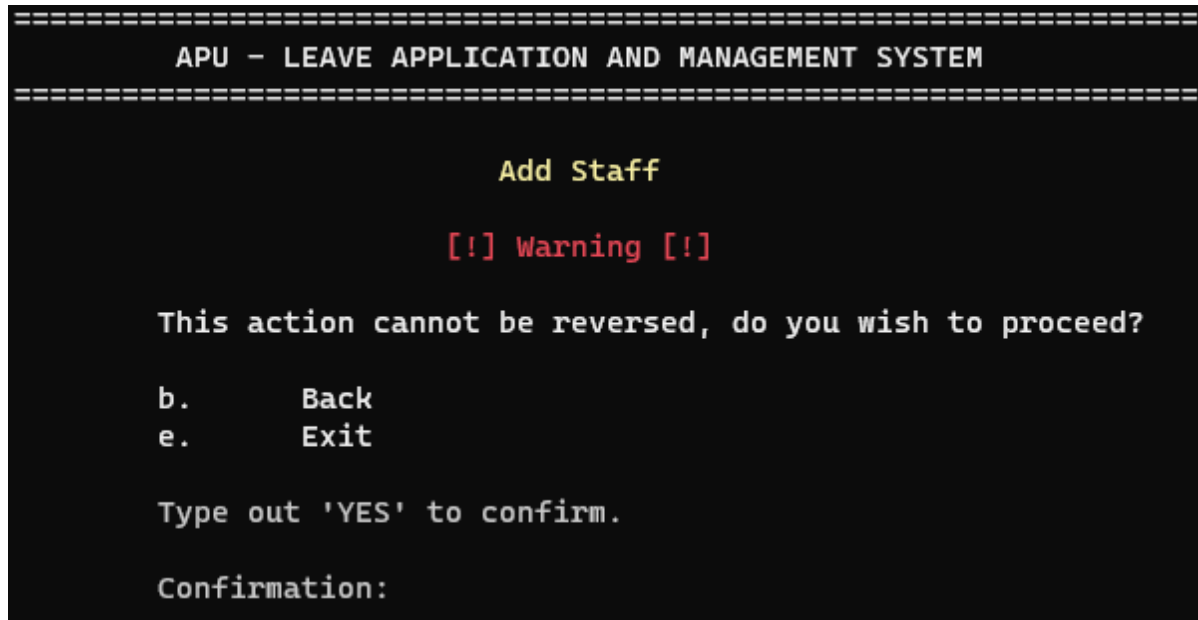


Figure 5.10.1: *Add staff menu confirmation*

The above figure is the add staff menu of the program. Due to limitations explained earlier, the system currently does not offer ways to remove staff members. Thus, the system informs the user that this action cannot be reversed and seeks further confirmation from the user. If adding a new staff is the user's intended action, the user has to type out 'YES' in the confirmation box as shown. The confirmation must be identical to the sample confirmation. Any deviation such as the 'YES' being typed out in lower case, misspelled, or with additional characters mixed in will not be accepted and the system will assume that confirmation is not given.

Upon a successful confirmation, the user will be asked to input the new staff member's username, staff ID, and password as shown in figure 5.10.2 below. By default, the maximum length for usernames, staff ID, and passwords are 50 characters.

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Add Staff

Confirmation successful.

Please enter the new staff's information.

Username: test

Staff ID: test1

Password: test123
```

Figure 5.10.2: *Input new username, staff ID, and password*

Afterwards, the user will be asked to input additional details such as the new staff member's department, rank, and their current leave balance through selection menus as shown in figure 5.10.3 below.

```
Department

1.      Administrative
2.      Management
3.      Academic
4.      Technical

Selection: 2

Staff Rank

1.      Staff
2.      Supervisor
3.      Administrator

Selection: 1
```

Figure 5.10.3: *Department and staff rank selection*

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Staff added.

Username:      test
Staff ID:      test1
Staff rank:    Staff
Department     Management
Leave balance:  20

Press enter to return to previous page.
```

Figure 5.10.4: *New staff information summary*

After successfully adding a new staff member, the user will be directed to this page where all the important information about the staff member is displayed. Pressing enter will return the user back to the admin menu.

5.11 Edit leave balance

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Edit Balance

1.      Edit current leave balance
2.      Step 1 year

b.      Back
e.      Exit

Selection:
```

Figure 5.11.1: *Edit leave balance menu*

The above figure 5.11.1 showcases the edit leave balance menu. This menu is accessible by selecting the “edit leave balance” option from the admin menu. Here, the user can choose to increment or decrement the current leave balance of all staff members. It also contains the more advanced function of stepping the system 1 year which will reset all leave balances for a new year.

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Edit current leave balance

This function adds or removes leave days from all
staff members.

1.      Add 1 day
2.      Remove 1 day

b.      Back

Selection: 1

Added 1 day.
```

Figure 5.11.2: *Added one day for all staff members*

The above figure is the leave balance increment or decrement menu. Actions on this menu is global and will affect the leave balances of all staff members. In figure 5.11.2, selection '1' was chosen and 1 day was added to all staff members' leave balances.

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Step 1 year

[!] Warning [!]

This function will reset all leave balances and is to be
used when a new year arrives. The following actions cannot
be reversed! Do you wish to continue?

Type out 'YES' to confirm.

Confirmation:
```

Figure 5.11.3: *Step 1 year confirmation*

The step 1 year function is intended to be used by the admin to reset all leave balances when a new year arrives. As this action cannot be reversed, it also seeks confirmation from the user as shown in figure 5.11.3 above.

```
Confirmation successful.  
  
Please enter the number of leave days for the new year.  
  
Number of leave days: 20
```

Figure 5.11.4: *Input default leave balance for new year*

Once a confirmation was successfully given, the system will request the user to input the default leave balance for the new year and will apply to all staff members. Due to limitations of the system and the uncertainty regarding leave balance carry forward policy of the company, none of the staffs' remaining leave balances will be carried forward into the new year. All leave balances will be reset to the default inputted by the admin. This function is shown in figure 5.11.4 above.

```
Leave balances updated.  
  
Press enter to return to previous page.
```

Figure 5.11.5: *Leave balance updated confirmation message*

Upon a successful system transition into the new year, the above message will be displayed to the admin.

5.12 View staff information

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Staff information

Please enter a staff ID.

Staff ID: staff2
```

Figure 5.12.1: *Staff information lookup ID input*

After selecting the staff information option from the admin menu, the user will be directed to this menu where they are prompted to input the ID of the staff who they wish to view the information of as shown in figure 5.12.1 above.

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

      Staff information

Please enter a staff ID.

Staff ID: staff2

      Staff Info
Username:      staff
Staff ID:      staff2
Staff rank:    Staff
Department     Management
Leave balance:  15

No.   Leave Type   Leave Date   Status
1     Compassionate 6-5-2023     Approved
2     Compassionate 7-5-2023     Approved
3     Compassionate 3-5-2028     Rejected
4     Medical leave  2-3-2023     Pending
5     Medical leave  5-3-2023     Pending
6     Annual leave   3-3-2023     Pending

Press enter to return to previous menu.
```

Figure 5.12.2: *Staff information display*

Upon inputting the staff ID, all of the relevant information about the staff member will be displayed as shown in figure 5.12.2. In addition, the staff member's leave list will also be displayed below along with the leave type, date, and statuses.

5.13 Generate monthly report

```
=====
APU - LEAVE APPLICATION AND MANAGEMENT SYSTEM
=====

Generate Report

This function will generate a monthly report txt file based on the
year and month you have entered, do you wish to continue?
y.      Yes

b.      Back
e.      Exit

Selection: y

Year: 2023

Month: 3
```

Figure 5.13.1: *Input year and month of monthly report*

Lastly, users of the admin rank also have the ability to generate a monthly leave report. By selecting the “generate monthly report” option from the admin menu, the user is directed to the menu as shown in figure 5.13.1.

First, the menu gives the user a brief explanation on the function and seek confirmation. Upon confirming the selection, the user is prompted to input the year and month of which a report will be generated for. Then, the user will be asked to select a department as well. Afterwards a report in the form of a .txt file will be generated.

```
Please choose a department.

1. Administrative
2. Management
3. Academic
4. Technical

Selection: 2

Report file successfully generated!
```

Figure 5.13.2: *File successfully generate message*

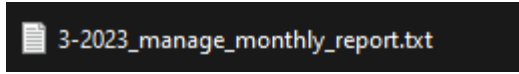


Figure 5.13.3: *Monthly report generated*

The name of the generated monthly report will begin with the month, followed by the year, and department keyword. For example, the monthly report text file shown in figure 5.13.3 is the monthly report for the March of 2023 for the management department.

```
3-2023 monthly staff leave report.
```

No.	Username	Staff ID	Rank	Type	Date	Status
1	staff	staff2	Staff	Medical leave	2-3-2023	Pending
2	staff	staff2	Staff	Medical leave	5-3-2023	Pending
3	staff	staff2	Staff	Annual leave	3-3-2023	Pending
4	staff	staff3	Staff	Compassionate	3-3-2023	Pending

Figure 5.13.4: *Contents of monthly report text file*

The above figure 5.13.4 is an example of the contents of the generated monthly report text file. At the top of the file, the month and year of the report is specified. Then, a list of the staff and their leave dates in the specified month, along with their statuses are printed below.

6.0 Conclusion

In summary, this program is considered a success as it achieves all its original objectives. The program sorts users into one of the three user categories upon login to the system based on the users' ranks. The ranks are the staff rank, supervisor rank, and admin rank, each with their own unique functions.

The staff rank grants the user basic functions such as applying for leave, cancelling existing leave, and view your personal leave information. These functions are also accessible to users of the supervisor and admin ranks. Supervisors can approve or reject leave applications as well as date lookup for staff members in their department. Lastly, admins can add new staff members, edit leave balance, lookup staff information, and generate monthly reports.

Unfortunately, there are also multiple limitations of the program that were unable to be addressed within the timeframe of the assignment. Limitations such as the inability to remove staff, edit leave balances of individual staff, and lack of checks on the day of the week of which a leave day lands on may be an inconvenience to some users. However, none of the limitations compromise the core functionality of the program. Solutions to many of these problems has already been identified but were not implemented due to time constraints.

In conclusion, given more time, experience, and knowledge in C programming, a more efficient and feature complete leave application can certainly be made. However, as it stands, the program still achieves all of its intended functions.

Throughout this course and assignment, my knowledge on the C programming language has expanded greatly. For a programming language that plays such a significant role in our modern society, I am humbled by how little I knew of it. I am honored to be given this opportunity to learn about C language and will strive to increase my proficiency in the language into the future.

7.0 References

1. The Urban Penguin. (2016, September 17). *Adding Color to Your Output From C - The Urban Penguin*. The Urban Penguin. <https://www.theurbanpenguin.com/4184-2/>
2. Das, R. (2020). *How to write a program in C to create a new file with name taken from a user*. Quora. Retrieved from: <https://www.quora.com/How-do-I-write-a-program-in-C-to-create-a-new-file-with-a-name-taken-from-the-user>
3. mehr20. (2015, January 9). *read text file and store in array in c programming*. Stack Overflow. <https://stackoverflow.com/questions/27856886/read-text-file-and-store-in-array-in-c-programming>
4. *C Program: Read the file and store the lines into an array - w3resource*. (2022, August 19). W3resource. <https://www.w3resource.com/c-programming-exercises/file-handling/c-file-handling-exercise-4.php>
5. *C Files I/O: Opening, Reading, Writing and Closing a file*. (2023). Programiz.com. <https://www.programiz.com/c-programming/c-file-input-output>
6. seg.server.fault. (2009, September 18). *How to get the date and time values in a C program?* Stack Overflow. <https://stackoverflow.com/questions/1442116/how-to-get-the-date-and-time-values-in-a-c-program>
7. *C Pointers (With Examples)*. (2023). Programiz.com. <https://www.programiz.com/c-programming/c-pointers>
8. cseprojects 14batch. (2022, February 5). *gets vs fgets for overflow in program*. Stack Overflow. <https://stackoverflow.com/questions/70994433/gets-vs-fgets-for-overflow-in-program>

8.0 Appendix

8.1 Backup Source Code Download

The following contains a GitHub link to the repository created for this individual assignment. It contains a backup of the source code as well as offering an alternate way of viewing the source code.

Link: <https://github.com/Lee-ZhiXuan/Leave-Application-System>