

**Schedule Management System**

**by**

**Student Shegufa Taranjum (162-35-1660)**

A project (SWE 231) submitted in fulfillment of the requirements for the degree of BSc in Software Engineering

**Department of Software Engineering**

**DAFFODIL INTERNATIONAL UNIVERSITY**

**Spring – 2017**

**DAFFODIL INTERNATIONAL UNIVERSITY**

DECLEARATION OF PROJECT

|  |  |
| --- | --- |
| Author Full Name | : Shegufa Taranjum |
| Date of Birth | : 25 January, 1997 |
| Title of Project | : Schedule Management System |
| Academic Session | : 2017 |
|  |  |

|  |  |
| --- | --- |
|  | Certified by: |
| SIGNATURE | SIGNATURE OF SUPERVISOR |
|  |  |
| STUDENT ID | NAME OF SUPERVISOR |
|  |  |
| Date: | Date: |

**Acknowledgement**

First of all I would like to thanks our respected teacher Dr. Md. ASRAF ALI. He is our course teacher. He was explained all the materials I use in this project. He covered all basic and advance concept in C programming. This is a project where we are being tested about our programming knowledge. There are lot of project available but I think this project is perfect for me. Because here I can apply all I have learn during the course.

My project is based on calendar and schedule management system. I feel the necessity of this project from my own life. I think this project will help all type of people. And also in this project I can apply properly what I learned during the whole course. Like pointer, file, data structures and lot more. This project make me more confident. Now I can apply programming knowledge to solve real problems. During completing the project I face lot of problem and this problem gain more knowledge.

I have also a plan to make this apps for android platform.

Thank you.

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROJECT DECLARATION | | | | i |
| ACKNOWLEDGEMENT | | | | ii |
| TABLE OF CONTENTS | | | | iii |
| LIST OF TABLES | | | | iv |
| LIST OF FIGURES | | | | v |
| LIST OF ABBREVIATIONS | | | | vi |
| LIST OF SYMBOLS | | | | vii |
| ABSTRACT | | | | Viii |
| CHAPTER 1 INTRODUCTION | | | | 1 |
|  | 1.1 | Background | | 2 |
|  | 1.2 | Motivation of the Project | | 3 |
|  | 1.3 | Problem Statement | | 5 |
|  | 1.4 | Project Objectives | | 6 |
|  | 1.5 | Project Organization | | 6 |
| CHAPTER 2 PROJECT METHODOLOGY | | | | 7 |
|  | 2.1 |  | | 10 |
|  | 2.2 |  | | 12 |
|  | 2.3 |  | | 15 |
| CHAPTER 3 RESULTS AND DISCUSSION | | | | 16 |
|  | 3.1 |  | | 16 |
|  | 3.2 |  | | 18 |
|  | 3.3 |  | | 18 |
|  | 3.4 |  | | 20 |
| CHAPTER 4 CONCLUSIONS AND RECOMMENDATIONS | | | | 20 |
|  | 4.1 | Findings and Contributions | | 24 |
|  | 4.2 | Recommendations for Future Works | | 25 |
| REFERENCES | | | | 27 |
| APPENDIX A | | |  | 30 |
| APPENDIX B | | |  | 31 |
|  | | |  |  |

**List of Tables:**

**List of Figures:**

**List of Abbreviation:**

**List of Symbols:**

**Abstract:**

**There** is nothing more important than time. Managing time is very important. There goes a proverb “Time and Tide waits for none”. We can change our life my managing it properly. Without making a schedule we cannot make best use of time.

My project is based on time, calendar, schedule and vacations. Using this software one can design their daily life more appropriately. They can utilize a single piece of time. Life is too short but there are lot of thing to do. We can do a lot of thing by only managing our times.

My whole paper will describe how this project was made and how to use this software. This is a quick overview of my project and below I describe the basic workflow of the project.

People see calendar, make their schedule. If it is a holiday schedule must be change. In my project this things are main object. One can make their own schedule and make best use of time by using this small software project.

**Chapter 1**

**INTRODUCTION**

* 1. **Background:** Time is very important thing in our life. We have to manage our time if we want to do a lot more thing in a small life. Managing time is very important. In our daily life there are lot more things we have to do. If we piece them and register this task with a fixed amount of time our job will done easily and more importantly all things we are done will organize. So we can check our progress too.
  2. **Motivation of the Project:** Time is important and we have to know how to manage our time for best use of it. I have to do a lot more thing in daily. And I always forget to do some work. Even I forget the date too. I think if I have make a software project which can help us to remember my tasks and also calendar and vacations it will be more fun. I think that’s not only my problem. There are several people who need this software. Like students, teacher, businessman, doctor all of the people can take advantage from this application software.
  3. **Problem Statement:** This is Schedule Management System. Its main task is to view a calendar according to current time and date. There should be command line interface. From where we can traverse calendar with commands. We can go next, previous, we can jump and we can go back to menu by command. Another part of this project is Schedule management. Actually this is the main part. Here people can add, delete, search and check their schedule or task. There are another criteria too. This project should be able to display vacation list according to Govt. vacation list. We can check vacation of current month, we can check all active vacation, inactive vacation etc.
  4. **Project Objectives:**
* Calendar
* Command line Interface to Control the Calendar
* Schedule Management System
* Add, Delete and View Schedule
* Upcoming Vacations
* All Active Vacations
* Search Vacations by Month
  1. **Project Organization**

**Chapter 2**

**PROJECT METHODOLOGY**

**2.1 Calendar:** Calendar is the first part of the available three parts. I use several functions. Let me explain those functions which I used to control my calendar.

2.1.1 GetCurrentTime : This function will create an template of Times structure and capture the current time from the system.

2.1.2 IsLeapYear: This Function will check whether the given year is leap year or not.

2.1.3 DayCode: This function has three parameter. It take day, month and year as its argument. Then calculate the day code according Georgian Calendar System and then return the code which is necessary for further use.

2.1.4 DisplayCalendar: This function also take day, month and year as its argument. Then calculate the day code using the previous function I have discussed. And then print a calendar to the system screen.

2.1.5 ConstructCalendar: This is very important function. Because every time when we get back to our main activity we call this function and it normally construct a new calendar according to the current time and date.

2.1.6 Next: This function is print the next month to the screen.

2.1.7 Previous: This function is print the previous month to the screen

2.1.8 Reset: This function will reset the whole calendar system and construct it again with current time and date

2.1.9 Jump: This function will help the user jump throw a long distance of calendar. On can view calendar any month of 1997 by using this jump function.

2.1.10 FullCalendar: This function will take input a year as its argument then print the full calendar of this year.

2.1.11 CalendarControl: I previously mentioned that my calendar is based on command line interface. All command and its control goes here. This function is responsible to handle commands.

**2.2 Linked List:** I use linked list for my project to store schedule data and operate on it. I have write some function for linked list. Like

2.2.1 CreateNode: This function will take an argument of Schedule type data and then allocate memory for it then create a node and return.

2.2.2 Add: This function also take an argument of Schedule type and add schedule to the list.

2.2.3 Delete: This function will delete the schedule from the list

2.2.4 Display: This function will display the list of the schedule.

**2.3 Schedule:** There are several task on schedule. So I have created some functions to manage the schedule operations. I have used two linked list one to store active data set and one to store completed tasks.

2.3.1 AddSchedule: This function will add a new schedule to the active list.

2.3.2 Upnext: this function will print the next task to the screen. It will calculate next task based on the current time and it’s expire time. Which task is near to current time this is the next task.

2.3.3 MarkDone: This function mark a list item as it is complete. When one item is marked as done it will add to the completed list.

2.3.4 DisplayActiveList: This function will show the user all active task according expire date.

2.3.5 DisplayCompleteList: This function will display all completed task.

**2.4 Vacations:** I have another feature called vacations. Now I am going to describe functions related to vacations.

2.4.1 ReadVacations: There is a data file which contain all vacation list and date. This function will read the file and store all data to the data structure in our case which is linked list.

2.4.2 SearchVacationByMonth: This function will take a month as argument. Then print all vacation available in that particular month.

2.4.3 DisplayUpNextVacation: This function will print the next vacation according to the current date.

2.4.4 DisplayAllAvailableVacation: This function will display all vacations after the current date which are not yet complete.

2.4.5: DisplayAllVacation: This function will display full vacation.

**2.5 File:** In my project there are several task which I have to write an external file. I have to write all active schedule and also completed schedule. And when the program starts working the program load data from the file. That’s how all the data I saved in my project.

**2.6 Others:** There are so many other helper function exits. Which will help to design the display, or help to interact with each other.

**CHAPTER 3**

**RESULT AND DISCUSSION**

**3.1 Menus:** There are 5 menus exist on the screen when anyone launch this program. The menus are, Calendar, Schedule, Vacations, Help and Exit. I will describe all the feature.

**3.2 Calendar:** This is one and only command interface in this project. It means we have to put command if we want to operate on the calendar. There are several command. I am listing all commands and its functionality here.

**3.2.1 next:** This command help me to go to next month.

**3.2.2 prev:** This command help me to go to previous month.

**3.2.3 jump:** This command help me to go to jump throw the calendar.

**3.2.4 full:** This command help me to view the whole calendar of a specific year.

**3.2.5 reset:** This command reset the calendar to the current time and date again.

**3.2.6 back:** This command will back to the main menu.

**3.3 Schedule:** There are total 6 options available under the schedule menu.

1. Create New Schedule: By pressing 1 user are triggered to creating a schedule and must have to provide a title, a description and expire date of the task. If everything ok this will create a task or schedule and add this to active list.

2. Up Next Schedule: By pressing 2 user can see the next task according to expire date of the task.

3. Mark As Done: If user press 3 he or she will see a list of active task. They can mark an item as done by providing its index number. After done an item as done it will listed to the completed list and remove from the active list.

4. View Active List: By pressing 4 user can check all active task.

5. View Done List: By pressing 5 user can check all completed list.

0. Back to Main Menu: By pressing 0 user will back again to the main menu.

**3.4 Vacations:** There are total 5 options available under the vacation menu.

1. Upcoming Vacation: By pressing 1 user can see the next upcoming vacation.

2. Search Vacation by Month: By pressing 2 User will able to search vacations according to month. All available list will display.

3. Available Vacations: If user press 3 they will see all available vacations in this year.

4. All Vacations: By pressing 4 user will able to see all the vacations in this particular year.

0. Back to Main Menu: Pressing 0 the system will prevent the user do perform any other operation and get back to the main menu.

**3.5 Help:** This section is the basic guide for users. How to user use my apps I will define here.

**3.6 Exit:** By pressing on this menu Exit user will terminate the program and will not able to do any perform before starts the program again. When user exit from the program all data are being write to the data file. So it’s important to exit properly from the system.

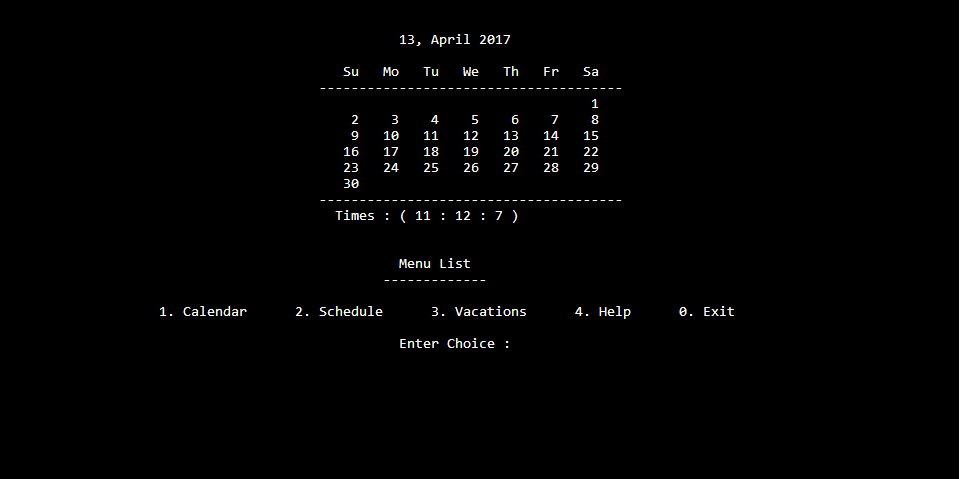
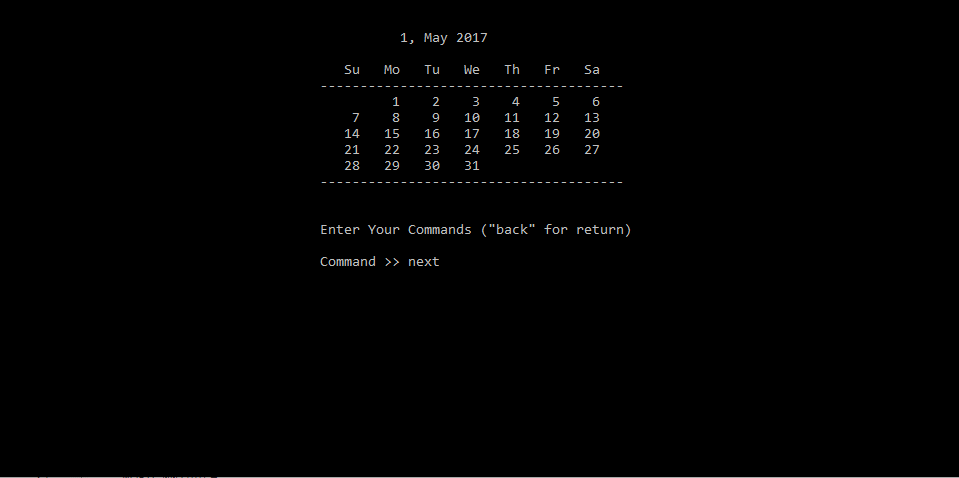
**** Figure 1: main menu

Figure 2: command prompt

****

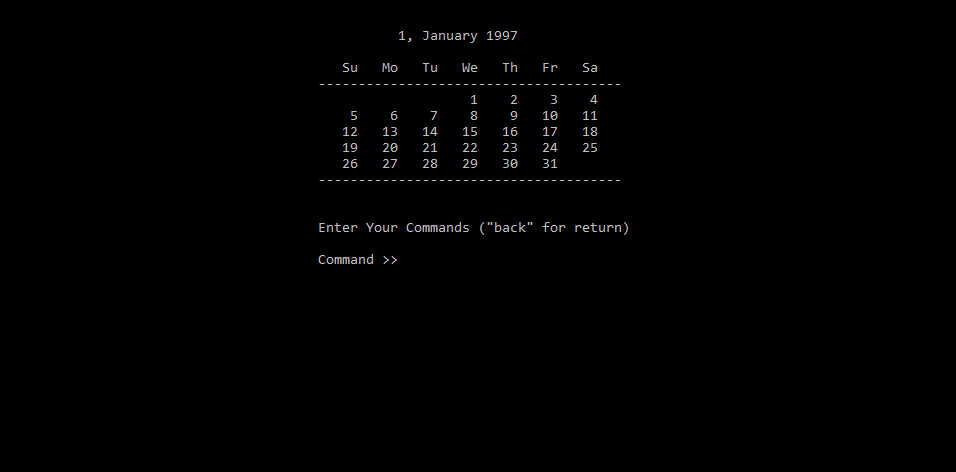
**** Figure 3: jumping to date jan 1997

Figure 4: viewing full calendar of 2017

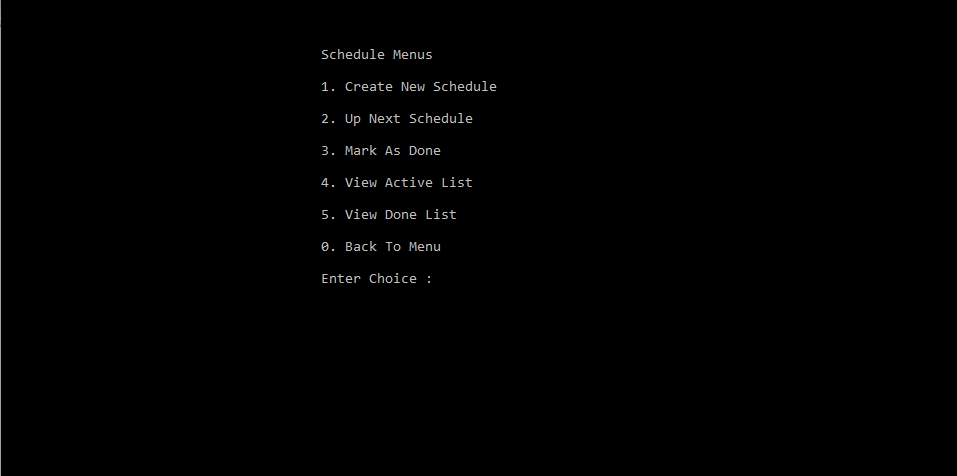
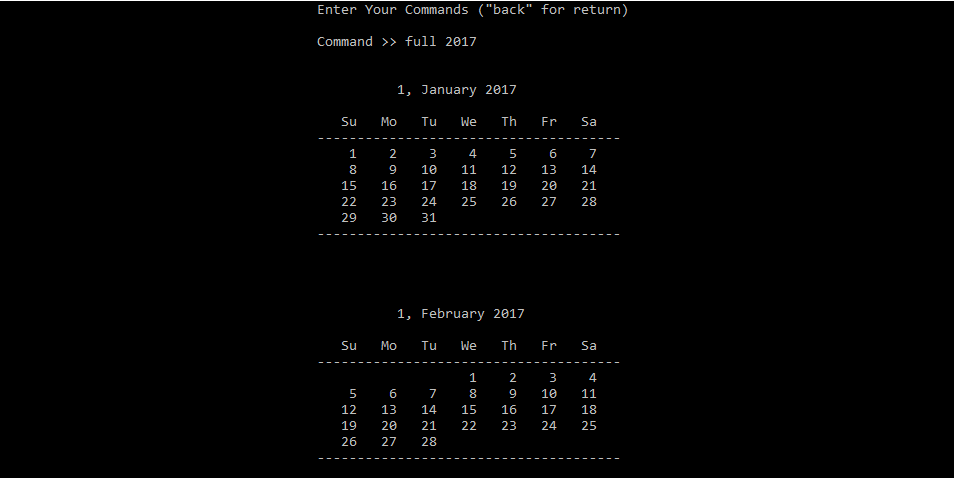
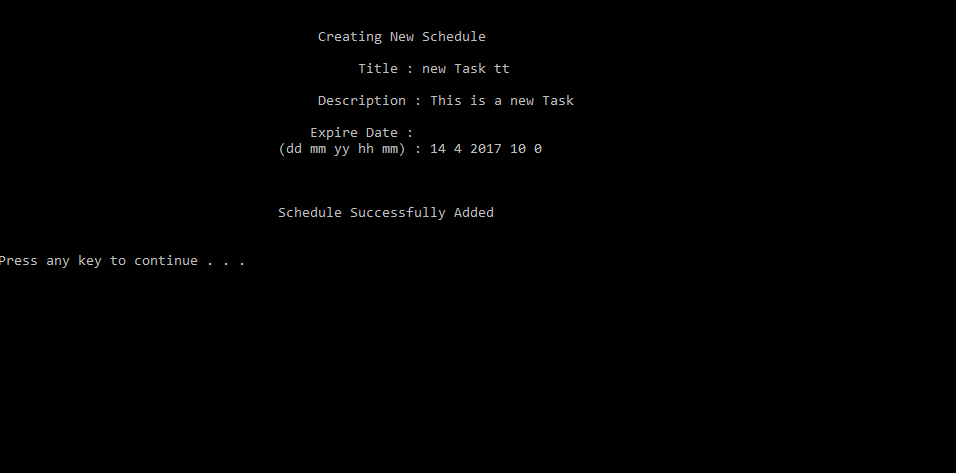
**** Figure 5: Schedule menus

Figure 6: creating a new schedule

****

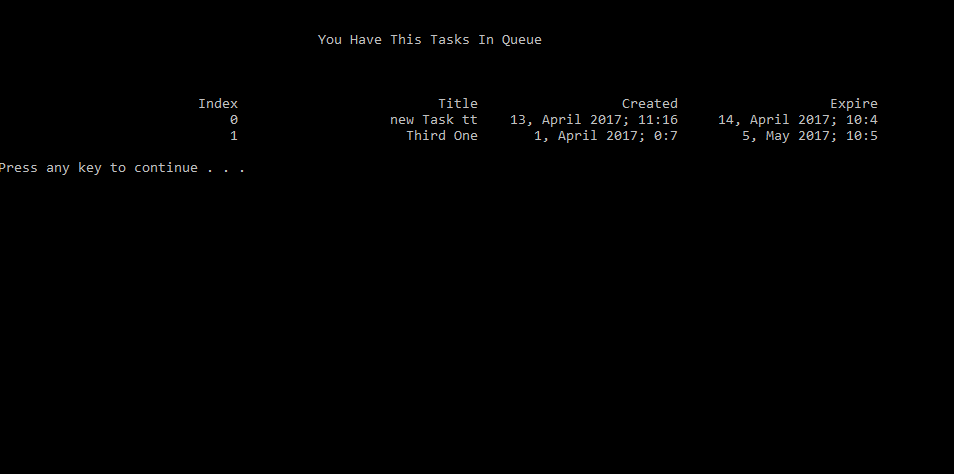
**** Figure 7: viewing active list

Figure 8: Vacation menus

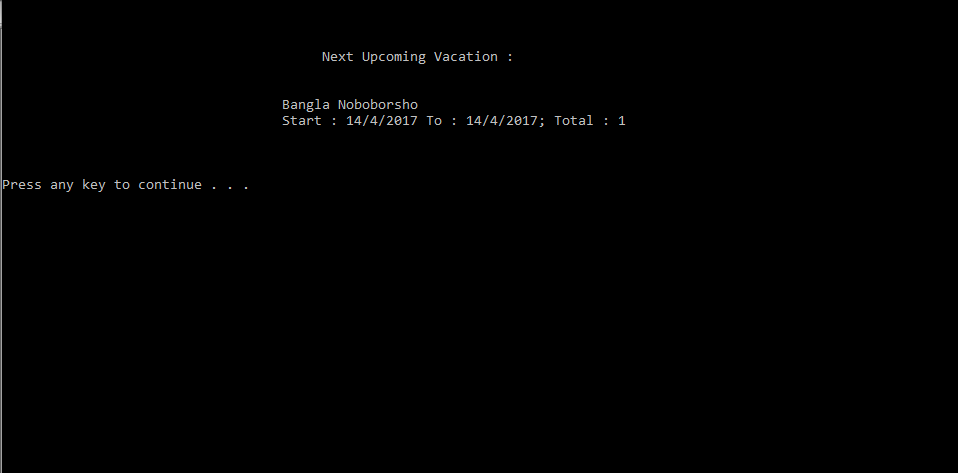
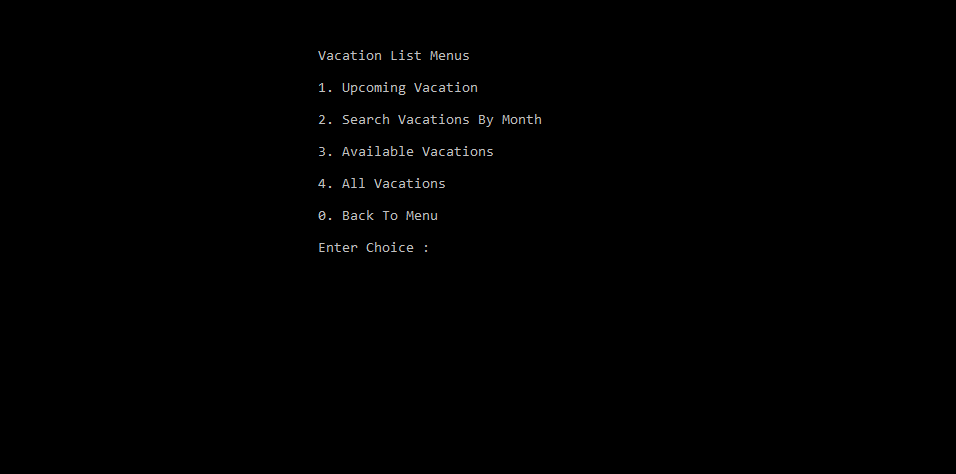
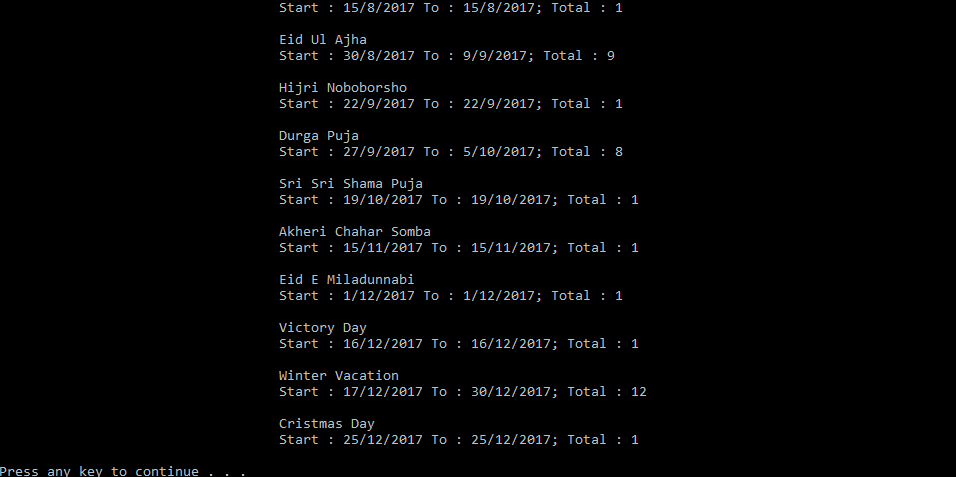
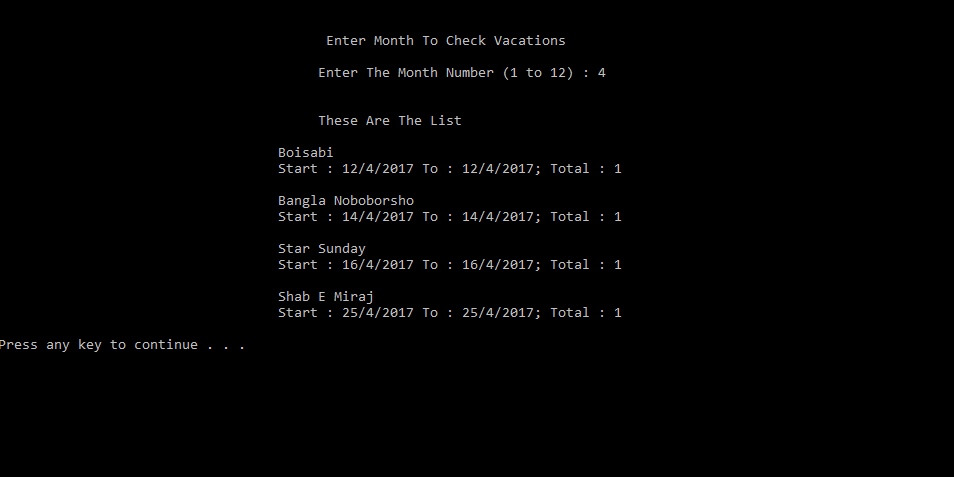
**** Figure 9: Next upcoming vacation

Figure 10: Search result of vacation

****  Figure 11: Full Vacation list

**CHAPTER 4**

**CONCLUTIONS AND RECOMMANDATIONS**

**4.1 Findings and Contributions**

**4.2 Recommendations for future work**

* **I will add notification to my project**
* **I will add weather system to my apps**
* **I will add an smart clock with alarm**
* **I will make this apps for android too**

**Appendix A**

**My code goes here…**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <time.h>**

**#include <string.h>**

**#include <windows.h>**

**#include <stdbool.h>**

**#define VACATIONMAX 50**

**int monthDays[13] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};**

**char monthNames[13][15] = {" ", "January", "February", "March",**

**"April", "May", "June", "July", "August",**

**"September", "October", "November", "December"};**

**struct times**

**{**

**WORD wYear;**

**WORD wMonth;**

**WORD wDayOfWeek;**

**WORD wDay;**

**WORD wHour;**

**WORD wMinute;**

**WORD wSecond;**

**WORD wMiliSecond;**

**};**

**typedef struct times Times;**

**struct date**

**{**

**int day;**

**int month;**

**int year;**

**};**

**typedef struct date VacationDate;**

**struct schedule**

**{**

**char title[30];**

**char description[100];**

**Times entry;**

**Times expire;**

**};**

**typedef struct schedule Schedule;**

**struct node**

**{**

**Schedule task;**

**struct node \*next;**

**};**

**typedef struct node List;**

**struct vac**

**{**

**char vacationName[100];**

**int total;**

**VacationDate startDate;**

**VacationDate endDate;**

**};**

**typedef struct vac Vacation;**

**Times getCurrentTime();**

**char\* getCalendarString(Times tm);**

**char\* getMonthNames(int month);**

**char\* getTimeString(Times tm);**

**bool isLeapYear(int year);**

**int dayCode(int day, int month, int year);**

**void displayCalendar(int day, int month, int year);**

**void constructCalendar();**

**void next();**

**void prev();**

**void reset();**

**void jump(int month, int year);**

**void displayFullCalendar(int year);**

**void calendarControl();**

**void menus();**

**void scheduleMenus();**

**List\* createNode(Schedule task);**

**void addToList(Schedule task);**

**void addToDone(Schedule task);**

**void removes(int index);**

**void displayList();**

**void displayDoneList();**

**Schedule createSchedule();**

**void addSchedule();**

**void upNext();**

**void markDone();**

**void displayDone();**

**void displayNotDone();**

**void writeDoneScheduleToFile();**

**void writeScheduleToFile();**

**void readDoneScheduleFromFile();**

**void readScheduleFromFile();**

**void readVacations();**

**void searchVacationByMonth();**

**void displayNextVacation();**

**void displayAvailableVacations();**

**void displayAllVacations();**

**char\* vacationVeiw(Vacation vac);**

**Vacation vacationList[VACATIONMAX];**

**int vacationCount = 0;**

**List \*head;**

**List \*done;**

**int currentYear;**

**int currentMonth;**

**int currentDay;**

**int listSize = 0;**

**int doneSize = 0;**

**int main()**

**{**

**int choice, subMenus;**

**readScheduleFromFile();**

**readDoneScheduleFromFile();**

**do**

**{**

**menus();**

**scanf("%d", &choice);**

**switch(choice)**

**{**

**case 0:**

**break;**

**case 1:**

**calendarControl();**

**break;**

**case 2:**

**do**

**{**

**system("CLS");**

**scheduleMenus();**

**scanf("%d", &subMenus);**

**switch(subMenus)**

**{**

**case 0:**

**writeScheduleToFile();**

**writeDoneScheduleToFile();**

**break;**

**case 1:**

**addSchedule();**

**break;**

**case 2:**

**upNext();**

**break;**

**case 3:**

**markDone();**

**break;**

**case 4:**

**displayNotDone();**

**break;**

**case 5:**

**displayDone();**

**break;**

**default:**

**printf("\n\n%40sInvalid Selections\n", " ");**

**break;**

**}**

**system("pause");**

**}while(subMenus != 0);**

**break;**

**case 3:**

**readVacations();**

**do**

**{**

**system("CLS");**

**vacationMenus();**

**scanf("%d", &subMenus);**

**switch(subMenus)**

**{**

**case 0:**

**break;**

**case 1:**

**displayNextVacation();**

**break;**

**case 2:**

**searchVacationByMonth();**

**break;**

**case 3:**

**displayAvailableVacations();**

**break;**

**case 4:**

**displayAllVacations();**

**break;**

**default:**

**printf("\n\n%40sInvalid Input\n\n");**

**break;**

**}**

**system("pause");**

**}while(subMenus != 0);**

**break;**

**}**

**}while(choice != 0);**

**return 0;**

**}**

**void menus()**

**{**

**int i;**

**system("CLS");**

**constructCalendar();**

**printf("\n\n%50sMenu List\n", " ");**

**printf("%48s", " ");**

**for(i=0; i<13; i++)**

**{**

**printf("-");**

**}**

**printf("\n\n%20s", " ");**

**printf("1. Calendar%6s2. Schedule%6s3. Vacations%6s4. Help%6s0. Exit\n", " ", " ", " ", " ");**

**printf("\n%50sEnter Choice : ", " ");**

**}**

**void scheduleMenus()**

**{**

**printf("\n\n\n%40sSchedule Menus\n\n", " ");**

**printf("%40s1. Create New Schedule\n", " ");**

**printf("\n%40s2. Up Next Schedule\n", " ");**

**printf("\n%40s3. Mark As Done\n", " ");**

**printf("\n%40s4. View Active List\n", " ");**

**printf("\n%40s5. View Done List\n", " ");**

**printf("\n%40s0. Back To Menu\n", " ");**

**printf("\n%40sEnter Choice : ", " ");**

**}**

**void vacationMenus()**

**{**

**printf("\n\n\n%40sVacation List Menus\n\n", " ");**

**printf("%40s1. Upcoming Vacation\n", " ");**

**printf("\n%40s2. Search Vacations By Month\n", " ");**

**printf("\n%40s3. Available Vacations\n", " ");**

**printf("\n%40s4. All Vacations\n", " ");**

**printf("\n%40s0. Back To Menu\n", " ");**

**printf("\n%40sEnter Choice : ", " ");**

**}**

**Times getCurrentTime()**

**{**

**Times tm;**

**GetSystemTime(&tm);**

**tm.wHour = tm.wHour+6;**

**return tm;**

**}**

**char\* getCalendarString(Times tm)**

**{**

**char dates[20];**

**sprintf(dates, "%d, %s %d; %d:%d", tm.wDay, getMonthNames(tm.wMonth), tm.wYear, tm.wHour, tm.wMinute);**

**return dates;**

**}**

**char\* getMonthNames(int month)**

**{**

**return monthNames[month];**

**}**

**char\* getTimeString(Times tm)**

**{**

**char times[20];**

**sprintf(times, "%d : %d : %d", tm.wHour, tm.wMinute, tm.wSecond);**

**return times;**

**}**

**bool isLeapYear(int year)**

**{**

**if(year % 4 == 0 && year % 100 != 0) return true;**

**if(year % 400 == 0) return true;**

**return false;**

**}**

**int dayCode(int day, int month, int year)**

**{**

**int y = year - (14-month) / 12;**

**int x = y + y/4 - y/100 + y/400;**

**int m = month + 12 \* ((14-month)/12) - 2;**

**int d = day + x + (31\*m) / 12 % 7;**

**return d;**

**}**

**void displayCalendar(int day, int month, int year)**

**{**

**int i, dcode;**

**if(isLeapYear(year))**

**{**

**monthDays[2] = 29;**

**}**

**dcode = dayCode(1, month, year);**

**printf("\n\n%50s", " ");**

**printf("%d, %s %d\n\n", day, getMonthNames(month), year);**

**printf("%40s", " ");**

**printf("%5s%5s%5s%5s%5s%5s%5s\n", "Su", "Mo", "Tu", "We", "Th", "Fr", "Sa");**

**printf("%40s", " ");**

**for(i=0; i<38; i++)**

**{**

**printf("-");**

**}**

**printf("\n%40s", " ");**

**for(i=0; i<dcode%7; i++)**

**{**

**printf("%5s", " ");**

**}**

**for(i=1; i<=monthDays[month]; i++)**

**{**

**printf("%5d", i);**

**if((i+dcode)%7 == 0)**

**{**

**printf("\n%40s", " ");**

**}**

**}**

**printf("\n%40s", " ");**

**for(i=0; i<38; i++)**

**{**

**printf("-");**

**}**

**printf("\n");**

**}**

**void constructCalendar()**

**{**

**Times tm = getCurrentTime();**

**char timess[20];**

**strcpy(timess, getTimeString(tm));**

**currentDay = tm.wDay;**

**currentMonth = tm.wMonth;**

**currentYear = tm.wYear;**

**system("CLS");**

**displayCalendar(currentDay, currentMonth, currentYear);**

**printf("%42sTimes : ( %s )\n", " ", timess);**

**}**

**void reset()**

**{**

**Times tm = getCurrentTime();**

**currentDay = tm.wDay;**

**currentMonth = tm.wMonth;**

**currentYear = tm.wYear;**

**system("CLS");**

**displayCalendar(currentDay, currentMonth, currentYear);**

**}**

**void next()**

**{**

**currentMonth = currentMonth + 1;**

**if(currentMonth > 12)**

**{**

**currentMonth = 1;**

**currentYear = currentYear + 1;**

**}**

**system("CLS");**

**displayCalendar(1, currentMonth, currentYear);**

**}**

**void prev()**

**{**

**currentMonth = currentMonth - 1;**

**if(currentMonth < 1)**

**{**

**currentMonth = 12;**

**currentYear = currentYear - 1;**

**}**

**system("CLS");**

**displayCalendar(1, currentMonth, currentYear);**

**}**

**void jump(int month, int year)**

**{**

**system("CLS");**

**displayCalendar(1, month, year);**

**}**

**void displayFullCalendar(int year)**

**{**

**int i;**

**for(i=1; i<=12; i++)**

**{**

**displayCalendar(1, i, year);**

**printf("\n\n");**

**}**

**}**

**void calendarControl()**

**{**

**int month, year;**

**char command[10];**

**system("CLS");**

**displayCalendar(currentDay, currentMonth, currentYear);**

**do**

**{**

**printf("\n\n%40sEnter Your Commands (\"back\" for return)\n\n", " ");**

**printf("%40sCommand >> ", " ");**

**scanf("%s", command);**

**if(strcmp(command, "next") == 0)**

**{**

**next();**

**}**

**else if(strcmp(command, "prev") == 0)**

**{**

**prev();**

**}**

**else if(strcmp(command, "jump") == 0)**

**{**

**scanf("%d %d", &month, &year);**

**jump(month, year);**

**}**

**else if(strcmp(command, "full") == 0)**

**{**

**scanf("%d", &year);**

**displayFullCalendar(year);**

**}**

**else if(strcmp(command, "reset") == 0)**

**{**

**reset();**

**}**

**else if(strcmp(command, "back") == 0)**

**{**

**break;**

**}**

**else**

**{**

**printf("%40sNot a Valid Command. See Help Section\n", " ");**

**}**

**}**

**while(strcmp(command, "back") != 0);**

**}**

**List\* createNode(Schedule task)**

**{**

**List \*newNode;**

**newNode = (List\*)malloc(sizeof(List));**

**if(newNode == NULL)**

**{**

**return 0;**

**}**

**newNode->task = task;**

**newNode->next = NULL;**

**return newNode;**

**}**

**void addToList(Schedule task)**

**{**

**char expOld[25], expNew[25];**

**List \*newNode = createNode(task);**

**if(head == NULL)**

**{**

**head = newNode;**

**head->next = NULL;**

**listSize++;**

**return;**

**}**

**strcpy(expOld, getCalendarString(head->task.expire));**

**strcpy(expNew, getCalendarString(task.expire));**

**List \*temp = head;**

**if(strcmp(expOld, expNew) == 1)**

**{**

**head = newNode;**

**head->next = temp;**

**listSize++;**

**return;**

**}**

**while(temp->next != NULL)**

**{**

**strcpy(expOld, getCalendarString(temp->next->task.expire));**

**if(strcmp(expOld, expNew) == 1)**

**{**

**break;**

**}**

**else**

**{**

**temp = temp->next;**

**}**

**}**

**newNode->next = temp->next;**

**temp->next = newNode;**

**listSize++;**

**return;**

**}**

**void addToDone(Schedule task)**

**{**

**List \*newNode = createNode(task);**

**if(done == NULL)**

**{**

**done = newNode;**

**done->next = NULL;**

**doneSize++;**

**return;**

**}**

**List \*temp = done;**

**while(temp->next != NULL)**

**{**

**temp = temp->next;**

**}**

**temp->next = newNode;**

**doneSize++;**

**return;**

**}**

**void removes(int index)**

**{**

**if(index < 0 || index > listSize)**

**{**

**printf("Invalid Index\n");**

**return;**

**}**

**if(index == 0)**

**{**

**head = head->next;**

**listSize--;**

**return;**

**}**

**List \*temp = head;**

**while(index - 1 > 0)**

**{**

**temp = temp->next;**

**index--;**

**}**

**temp->next = temp->next->next;**

**listSize--;**

**}**

**void displayList()**

**{**

**List \*temp = head;**

**char start[25], end[25], title[30];**

**int index = 0;**

**printf("\n\n%20s%10s%30s%25s%25s\n", " ", "Index", "Title", "Created", "Expire");**

**while(temp != NULL)**

**{**

**strcpy(start, getCalendarString(temp->task.entry));**

**strcpy(end, getCalendarString(temp->task.expire));**

**strcpy(title, temp->task.title);**

**printf("%20s%10d%30s%25s%25s\n", " ", index++, title, start, end);**

**temp = temp->next;**

**}**

**printf("\n");**

**}**

**void displayDoneList()**

**{**

**List \*temp = done;**

**char start[25], end[25], title[30];**

**int index = 0;**

**printf("\n\n%20s%10s%30s%25s%25s\n", " ", "Index", "Title", "Created", "Expire");**

**while(temp != NULL)**

**{**

**strcpy(start, getCalendarString(temp->task.entry));**

**strcpy(end, getCalendarString(temp->task.expire));**

**strcpy(title, temp->task.title);**

**printf("%20s%10d%30s%25s%25s\n", " ", index++, title, start, end);**

**temp = temp->next;**

**}**

**printf("\n");**

**}**

**Schedule createSchedule()**

**{**

**char title[30], description[100];**

**Times entry, expire;**

**int dd, mm, yy, hr, min;**

**Schedule temp;**

**system("CLS");**

**getchar();**

**printf("\n\n%40sCreating New Schedule\n", " ");**

**printf("\n%40s%10s : ", " ", "Title");**

**gets(title);**

**printf("\n%40s%10s : ", " ", "Description");**

**gets(description);**

**printf("\n%35s%15s : ", " ", "Expire Date");**

**printf("\n%35s%15s : ", " ", "(dd mm yy hh mm)");**

**scanf("%d %d %d %d %d", &dd, &mm, &yy, &hr, &min);**

**expire.wDay = dd;**

**expire.wMonth = mm;**

**expire.wYear = yy;**

**expire.wHour = hr;**

**expire.wMinute = mm;**

**entry = getCurrentTime();**

**strcpy(temp.title, title);**

**strcpy(temp.description, description);**

**temp.entry = entry;**

**temp.expire = expire;**

**return temp;**

**}**

**void addSchedule()**

**{**

**Schedule temp = createSchedule();**

**addToList(temp);**

**printf("\n\n\n%35sSchedule Successfully Added\n\n\n", " ");**

**}**

**void upNext()**

**{**

**char entry[25], title[30], description[100], expire[25];**

**system("CLS");**

**if(head == NULL)**

**{**

**printf("\n\n\n%40sThere Is No Active Task\n\n", " ");**

**return;**

**}**

**strcpy(entry, getCalendarString(head->task.entry));**

**strcpy(expire, getCalendarString(head->task.expire));**

**strcpy(title, head->task.title);**

**strcpy(description, head->task.description);**

**printf("\n\n%40sYour Next Task\n\n", " ");**

**printf("%35s%15s %s\n", " ", "Title : ", title);**

**printf("%35s%15s %s\n", " ", "Created On : ", entry);**

**printf("%35s%15s %s\n", " ", "Expire On : ", expire);**

**printf("%35s%15s %s\n", " ", "Description : ", description);**

**printf("\n\n");**

**}**

**void markDone()**

**{**

**Schedule tempSchedule;**

**List \*temp = head;**

**int index;**

**system("CLS");**

**printf("\n\n%40sYour Schedule List", " ");**

**printf("\n%40sEnter Index to Mark Done", " ");**

**printf("\n\n%40sEnter -1 for Cancel Editing\n", " ");**

**displayList();**

**printf("\n\n%40sEnter Index : ");**

**scanf("%d", &index);**

**if(index == -1)**

**{**

**printf("\n%40sCancel Editing\n");**

**return;**

**}**

**if(index < 0 || index > listSize)**

**{**

**printf("\n%40sInvalid Index\n", " ");**

**return;**

**}**

**while(index - 1 > 0)**

**{**

**temp = temp->next;**

**index--;**

**}**

**strcpy(tempSchedule.title, temp->task.title);**

**strcpy(tempSchedule.description, temp->task.description);**

**tempSchedule.entry = temp->task.entry;**

**tempSchedule.expire = temp->task.expire;**

**addToDone(tempSchedule);**

**removes(index);**

**printf("\n\n%30sSchedule Successfully Marked As Done\n", " ");**

**}**

**void displayDone()**

**{**

**system("CLS");**

**printf("\n\n%40sYou Have Done This Tasks\n\n", " ");**

**displayDoneList();**

**}**

**void displayNotDone()**

**{**

**system("CLS");**

**printf("\n\n%40sYou Have This Tasks In Queue\n\n", " ");**

**displayList();**

**}**

**void readVacations()**

**{**

**FILE \*file;**

**char name[100];**

**int total, startD, endD, startM, endM;**

**Vacation temp;**

**VacationDate startDate, endDate;**

**file = fopen("vacations.dat", "r");**

**if(file == NULL)**

**{**

**exit(0);**

**}**

**while(true)**

**{**

**fgets(name, 255, (FILE \*) file);**

**if(strcmp(name, "ENDOFFILE") == 0)**

**{**

**break;**

**}**

**fscanf(file, "%d %d %d %d %d\n", &startD, &startM, &endD, &endM, &total);**

**startDate.day = startD;**

**startDate.month = startM;**

**startDate.year = 2017;**

**endDate.day = endD;**

**endDate.month = endM;**

**endDate.year = 2017;**

**strcpy(temp.vacationName, name);**

**temp.startDate = startDate;**

**temp.endDate = endDate;**

**temp.total = total;**

**vacationList[vacationCount++] = temp;**

**}**

**fclose(file);**

**}**

**void searchVacationByMonth()**

**{**

**int mon, i;**

**char vacView[250];**

**system("CLS");**

**printf("\n\n%40s Enter Month To Check Vacations\n\n", " ");**

**printf("%40sEnter The Month Number (1 to 12) : ", " ");**

**scanf("%d", &mon);**

**printf("\n\n%40sThese Are The List\n\n", " ");**

**for(i=0; i<vacationCount; i++)**

**{**

**if(vacationList[i].startDate.month == mon)**

**{**

**strcpy(vacView, vacationVeiw(vacationList[i]));**

**printf("%35s%s\n\n", " ", vacView);**

**}**

**}**

**}**

**void displayNextVacation()**

**{**

**int i;**

**char vacView[250];**

**Times tm = getCurrentTime();**

**system("CLS");**

**for(i=0; i<vacationCount; i++)**

**{**

**if(vacationList[i].startDate.month == tm.wMonth)**

**{**

**if(vacationList[i].startDate.day > tm.wDay)**

**{**

**strcpy(vacView, vacationVeiw(vacationList[i]));**

**break;**

**}**

**}**

**else if(vacationList[i].startDate.month > tm.wMonth)**

**{**

**strcpy(vacView, vacationVeiw(vacationList[i]));**

**break;**

**}**

**}**

**printf("\n\n\n%40sNext Upcoming Vacation : \n\n\n%35s%s\n\n\n\n", " ", " ", vacView);**

**}**

**void displayAvailableVacations()**

**{**

**int i;**

**char vacView[250];**

**Times tm = getCurrentTime();**

**system("CLS");**

**printf("\n\n%40s These Are Available Vacation List\n\n", " ");**

**for(i=0; i<vacationCount; i++)**

**{**

**if(vacationList[i].startDate.month > tm.wMonth)**

**{**

**strcpy(vacView, vacationVeiw(vacationList[i]));**

**printf("%35s%s\n\n", " ", vacView);**

**}**

**}**

**}**

**void displayAllVacations()**

**{**

**int i;**

**char vacView[255];**

**system("CLS");**

**printf("\n\n%40s Full Vacation List of 2017\n\n", " ");**

**for(i=0; i<vacationCount; i++)**

**{**

**strcpy(vacView, vacationVeiw(vacationList[i]));**

**printf("%35s%s\n\n", " ", vacView);**

**}**

**}**

**char\* vacationVeiw(Vacation vac)**

**{**

**char view[150], dat[100];**

**strcpy(view, vac.vacationName);**

**sprintf(dat , "%35sStart : %d/%d/%d To : %d/%d/%d; Total : %d", " ", vac.startDate.day, vac.startDate.month, vac.startDate.year,**

**vac.endDate.day, vac.endDate.month, vac.endDate.year, vac.total);**

**strcat(view, dat);**

**return view;**

**}**

**void readScheduleFromFile()**

**{**

**FILE \*file;**

**char title[30], description[100];**

**int startD, startM, startY, startH, startMn;**

**int endD, endM, endY, endH, endMn;**

**int slen;**

**Schedule temp;**

**Times start, end;**

**file = fopen("tasksdata.dat", "r");**

**if(file == NULL)**

**{**

**exit(0);**

**}**

**while(!feof(file))**

**{**

**fgets(title, 30, (FILE \*) file);**

**slen = strlen(title);**

**title[slen-1] = '\0';**

**fgets(description, 100, (FILE \*) file);**

**slen = strlen(description);**

**description[slen-1] = '\0';**

**fscanf(file, "%d %d %d %d %d %d %d %d %d %d\n", &startD, &startM, &startY, &startH, &startMn, &endD, &endM, &endY, &endH, &endMn);**

**start.wDay = startD;**

**start.wMonth = startM;**

**start.wYear = startY;**

**start.wHour = startH;**

**start.wMinute = startMn;**

**end.wDay = endD;**

**end.wMonth= endM;**

**end.wYear = endY;**

**end.wHour = endH;**

**end.wMinute = endMn;**

**strcpy(temp.title, title);**

**strcpy(temp.description, description);**

**temp.entry = start;**

**temp.expire = end;**

**addToList(temp);**

**}**

**fclose(file);**

**}**

**void writeScheduleToFile()**

**{**

**FILE \*file;**

**List \*temp = head;**

**file = fopen("tasksdata.dat", "w+");**

**if(file == NULL)**

**{**

**exit(0);**

**}**

**while(temp != NULL)**

**{**

**fprintf(file, "%s\n", temp->task.title);**

**fprintf(file, "%s\n", temp->task.description);**

**fprintf(file, "%d %d %d %d %d %d %d %d %d %d", temp->task.entry.wDay, temp->task.entry.wMonth, temp->task.entry.wYear, temp->task.entry.wHour, temp->task.entry.wMinute,**

**temp->task.expire.wDay, temp->task.expire.wMonth, temp->task.expire.wYear, temp->task.expire.wHour, temp->task.expire.wMinute);**

**fprintf(file, "%s", "\n");**

**temp = temp->next;**

**}**

**fclose(file);**

**}**

**void readDoneScheduleFromFile()**

**{**

**FILE \*file;**

**char title[30], description[100];**

**int startD, startM, startY, startH, startMn;**

**int endD, endM, endY, endH, endMn;**

**int slen;**

**Schedule temp;**

**Times start, end;**

**file = fopen("donetasksdata.dat", "r");**

**if(file == NULL)**

**{**

**exit(0);**

**}**

**while(!feof(file))**

**{**

**fgets(title, 30, (FILE \*) file);**

**slen = strlen(title);**

**title[slen-1] = '\0';**

**fgets(description, 100, (FILE \*) file);**

**slen = strlen(description);**

**description[slen-1] = '\0';**

**fscanf(file, "%d %d %d %d %d %d %d %d %d %d\n", &startD, &startM, &startY, &startH, &startMn, &endD, &endM, &endY, &endH, &endMn);**

**start.wDay = startD;**

**start.wMonth = startM;**

**start.wYear = startY;**

**start.wHour = startH;**

**start.wMinute = startMn;**

**end.wDay = endD;**

**end.wMonth= endM;**

**end.wYear = endY;**

**end.wHour = endH;**

**end.wMinute = endMn;**

**strcpy(temp.title, title);**

**strcpy(temp.description, description);**

**temp.entry = start;**

**temp.expire = end;**

**addToDone(temp);**

**}**

**fclose(file);**

**}**

**void writeDoneScheduleToFile()**

**{**

**FILE \*file;**

**List \*temp = done;**

**file = fopen("donetasksdata.dat", "w+");**

**if(file == NULL)**

**{**

**exit(0);**

**}**

**while(temp != NULL)**

**{**

**fprintf(file, "%s\n", temp->task.title);**

**fprintf(file, "%s\n", temp->task.description);**

**fprintf(file, "%d %d %d %d %d %d %d %d %d %d", temp->task.entry.wDay, temp->task.entry.wMonth, temp->task.entry.wYear, temp->task.entry.wHour, temp->task.entry.wMinute,**

**temp->task.expire.wDay, temp->task.expire.wMonth, temp->task.expire.wYear, temp->task.expire.wHour, temp->task.expire.wMinute);**

**fprintf(file, "%s", "\n");**

**temp = temp->next;**

**}**

**fclose(file);**

**}**

**REFERENCES**