

CSE102 Computer Programming HW7 Report

Part 1

In this part we were asked to write a Turkish draughts game code. I couldn't be able to write recursive move function ; my old homework wasn't working too. So I'm not changing my old homework.

```
36
37 void init_board(piece board[][8]){ /* board at the start*/
38 int i;
39     for(i=0; i<8;i++){
40         board[0][i]=empty;
41         board[1][i]=black_man;
42         board[2][i]=black_man;
43         board[3][i]=empty;
44         board[4][i]=empty;
45         board[5][i]=white_man;
46         board[6][i]=white_man;
47         board[7][i]=empty;
48     }
49
50 }
51 int check_end_of_game(piece board[][8]){ /*checks if game is over*/
52 piece check[64];
53 int a=0,whi=0, bl=0,res,i,j;
54     for(i=0;i<8;i++){
55         for(j=0;j<8;j++){
56             check[a]=board[i][j];
57             a++;
58         }
59     }
60
61     for(a=0;a<64;a++){
62         if(check[a]!=black_man && check[a]!=black_king){
63             whi++;
64         }else if(check[a]!=white_man && check[a]!=white_king){
65             bl++;
66         }
67     }
68
69     if(whi==64){ /*if there is no black pieces*/
70         res=white;
71     } else if(bl==64){ /*if there is no white pieces*/
72         res=black;
73     } else { /*if there are both on the board */
74         res =-1;
75     }
76     return res;
77 },
78 void display_board(piece board[][8]){
79 int i,j;
80
81     for(i=0;i<8;i++){
82         for(j=0;j<8;j++){
83             if(board[i][j]==empty){
84                 printf(" - ");
85             }else if(board[i][j]==black_man){
86                 printf(" b ");
87             }else if(board[i][j]==white_man){
88                 printf(" w ");
89             }else if(board[i][j]==white_king){
90                 printf(" W ");
91             }else if(board[i][j]==black_king){
92                 printf(" B ");
93             }
94         }
95         printf("\n");
96     }
97     printf("\n");
98 }
99 }
```

init_board function is initializing board.

check_end_of_game checks if a players pieces are doesn't exist on board; this means that player is lose.

display_board prints board

I'm not adding ss of sample game functions and move; I dont have a acceptable move function, so samples too.

```
ilkay@ubuntu: ~/Desktop
- - - - -
b b b b b b b b
b - b b b b b b
- b - - - - -
w w w w w w w w
w w w w w w w w
- - - - -
- - - - -
b b b b b b b b
b - b b b b b b
- - - - -
w b - - - - -
- w w w w w w w
w w w w w w w w
- - - - -
- - - - -
b b b b b b b b
b - b b b b b b
- - - - -
- - w - - - -
- w w w w w w w
w w w w w w w w
- - - - -
```

Part 2

This program takes 2 dates from user, generates other dates between taken dates and writes them to a file. Then finds weekday of generated days and changes their format and writes them to a new file.

```
71 }
72 int check_date(int day, int month, int year){ /*this function checks if date is invalid or not*/
73     if(year<0 || month<1 || month>12 || day<1){
74         return 0;
75     }else if((month==4 || month==6 || month==9 || month == 11) && day>30){
76         return 0;
77     }else if((month==1 || month==3 || month==5 || month==7 || month==8 || month==10 || month == 12) && day>31){
78         return 0;
79     }else if(month==2 && (year%400==0) && day>29){ /*leap years*/
80         return 0;
81     }else if(month==2 && (year%100==0) && day>28){
82         return 0;
83     }else if(month==2 && (year%4==0) && day>29){
84         return 0;
85     }else if(month==2 && (year%4!=0) && day>28){
86         return 0;
87     }else {
88         return 1;
89     }
90 }
91 int compare_date(int day, int month, int year, int day_f, int month_f, int year_f){ /*this function compares dates */
92     if((year_f<year) || ((year_f==year)&&(month_f<month)) || ((year_f==year)&&(month_f==month)&& (day_f<day)) ){
93         return 0;
94     }else {
95         return 1;
96     }
97 }
98 int is_leap_year(int year){ /*checks if given year is a leap year*/
99     if(year%400==0) {
100         return 1;
101     }else if(year%100==0){
102         return 0;
103     }else if(year%4==0){
104         return 1;
105     }else {
106         return 0;
107     }
108 }
109 int find_size(char str[]){ /* Finds the size of a string*/
110     int count;
111     for ( count = 0; str[count]!= 0; ++count); /* counts until NULL*/
112     return count;
113 }
114 void new_date() /*changes format of date*/{
115     int day, month, year, temp, temp2=0;
116 }
```

check_date is a function that controls if given date is invalid or not. For example if user enters 32s042554 it returns 0 which means date is not true.

compare_dates compares two date and returns 0 if first one is bigger, else returns 1.

is_leap_year checks if given year is a leap year or not.

find_size finds size of a string.

```

15 void generate_days(){
16 int day, month, year, day_f, month_f, year_f, flag=1;
17 char ch;
18 FILE *outp;
19 outp = fopen("input_date.txt", "w"); /*open file*/
20 char date_i[11], date_f[11], date_n[11]; /*strings for start and final dates*/
21 printf("Enter start date(dd/mm/yyyy):\n");
22 scanf("%s", date_i);
23 printf("Enter final date(dd/mm/yyyy):\n");
24 scanf("%s", date_f);
25 sscanf(date_i,"%d %c %d %c %d", &day, &ch, &month, &ch, &year); /* turn string to integer values */
26 sscanf(date_f,"%d %c %d %c %d", &day_f, &ch, &month_f, &ch, &year_f);
27 if((check_date(day, month, year) && check_date(day_f, month_f, year_f))!=1){
28     printf("Invalid date\n"); /* check date if its invalid ask again with recursion*/
29     flag=0;
30     generate_days();
31 }else if(compare_date(day, month, year, day_f, month_f, year_f)!=1){ /*compare dates; first one must be earlier*/
32     printf("Start date must be earlier than final date!\n");
33     flag=0;
34     generate_days();
35 }else{
36     printf("\n%s\n", date_i );
37     fputs(date_i, outp); /*write first date*/
38     putc('\n', outp);
39     while(flag){ /* change date until final date */
40         if((month==1 || month==3 || month==5 || month==7 || month==8 || month==10 || month == 12) && day == 31){
41             if(month==12){
42                 day=1;
43                 month=1;
44                 year++;
45             }else{
46                 day=1;
47                 month++;
48             }
49         }else if((month==4 || month==6 || month==9 || month == 11) && day==30){
50             day=1;
51             month++;
52         }else if((month==2) && (year%4==0) && day==29) {
53             day=1;
54             month++;
55         }else if((month==2) && (year%4!=0) && day==28){
56             day=1;
57             month++;
58         }else {
59             day++;
60         }
61         sprintf(date_n, "%02d%c%02d%c%02d", day,ch,month,ch,year ); /*take new date to string*/
62         printf("%s\n", date_n ); /*write new date to file and terminal*/
63         fputs(date_n, outp);
64         putc('\n', outp);
65         if(day==day_f && month==month_f && year==year_f){
66             flag=0;
67         }
68     }
69     fclose(outp); /*close file*/
70 }
71 }
72 int check_date(int day, int month, int year){ /*this function checks if date is invalid or not*/
73     if(year<0 || month<1 || month>12 || day<1){
74         return 0;

```

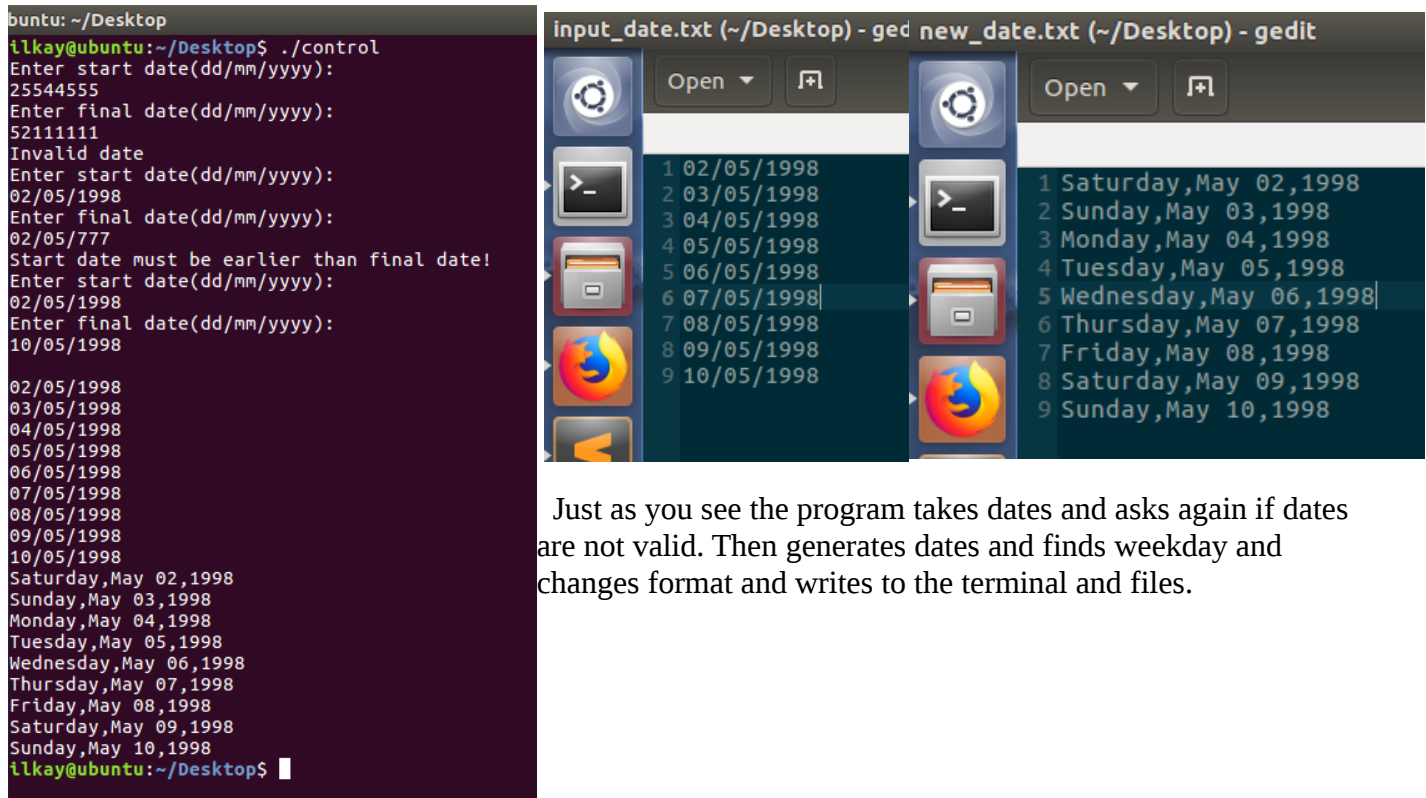
generate_days takes dates from user calls check_date and compare_dates functions to see if date are okay. Then until final date it generates dates and writes them to input_date.txt and terminal.

```

113 }
114 void new_date() /*changes format of date*/{
115     int day, month, year, temp, temp2=0;
116     char ch, comma=',';
117     char date[11], new_date[50];
118     int month_key[12]={1,4,4,0,2,5,0,3,6,1,4,6}; /*this code is written depending on key value formula*/
119     int year_code[4]={4,2,0,6};
120     FILE *inp;
121     FILE *outp;
122     inp = fopen("input_date.txt", "r");
123     outp = fopen("new_date.txt", "w");
124     while(fgets(date, 1024, inp)!=0){ /*read until end of file*/
125         sscanf(date, "%d %c %d %c %d", &day, &ch, &month, &ch, &year); /*turn string to int and char*/
126         temp2=year; /*year may change so temp2 will hold its first value*/
127         temp = (year%100)/4;
128         temp+=day;
129         temp+=month_key[month-1];
130         if((month==1 || month==2) && (is_leap_year(year)==1)){
131             temp--;
132         }
133         if(year>2099) { /*every 400 year repeats itself so we will subtract 400 to find year code */
134             while(year>2099){
135                 year = year-400;
136             }
137         }
138         if(year>=2000){
139             temp+=year_code[3];
140         }else if (year>=1900){
141             temp+=year_code[2];
142         }else if(year>=1800){
143             temp+=year_code[1];
144         }else if(year>=1700){
145             temp+=year_code[0];
146         }
147         temp+= temp2%100;
148         temp=temp%7;
149         if(temp==0){ /*write day to new date string*/
150             sprintf(new_date, "Saturday,");
151         }else if(temp==1){
152             sprintf(new_date, "Sunday,");
153         }else if(temp==2){
154             sprintf(new_date, "Monday,");
155         }else if(temp==3){
156             sprintf(new_date, "Tuesday,");
157         }else if(temp==4){
158             sprintf(new_date, "Wednesday,");
159         }else if(temp==5){
160             sprintf(new_date, "Thursday,");
161         }else {
162             sprintf(new_date, "Friday,");
163         }
164     }
165     if (month==1){ /*write month to new_date string*/
166         sprintf(new_date+find_size(new_date), "January ");
167     }else if(month==2){
168         sprintf(new_date+find_size(new_date), "February ");
169     }else if(month==3){
170         sprintf(new_date+find_size(new_date), "March ");
171     }else if(month==4){
172         sprintf(new_date+find_size(new_date), "April ");
173     }else if(month==5){
174         sprintf(new_date+find_size(new_date), "May ");
175     }else if(month==6){
176         sprintf(new_date+find_size(new_date), "June ");
177     }else if(month==7){
178         sprintf(new_date+find_size(new_date), "July ");
179     }else if(month==8){
180         sprintf(new_date+find_size(new_date), "August ");
181     }else if(month==9){
182         sprintf(new_date+find_size(new_date), "September ");
183     }else if(month==10){
184         sprintf(new_date+find_size(new_date), "October ");
185     }else if(month==11){
186         sprintf(new_date+find_size(new_date), "November ");
187     }else if(month==12){
188         sprintf(new_date+find_size(new_date), "December ");
189     }
190     sprintf(new_date+find_size(new_date), "%02d%c%02d", day, comma, temp2); /*add day of month and year to string*/
191     fputs(new_date, outp);
192     putc('\n', outp); /*write to the file*/
193     printf("%s\n", new_date); /*printf for terminal*/
194 }
195 fclose(inp);
196 fclose(outp); /*close files*/
197 }

```

new_date function finds weekday of dates in input_date.txt file. This function uses key value formula to find weekday. Then changes format of date and writes to terminal and new_date.txt.



```
buntu: ~/Desktop
ilkay@ubuntu:~/Desktop$ ./control
Enter start date(dd/mm/yyyy):
25544555
Enter final date(dd/mm/yyyy):
52111111
Invalid date
Enter start date(dd/mm/yyyy):
02/05/1998
Enter final date(dd/mm/yyyy):
02/05/777
Start date must be earlier than final date!
Enter start date(dd/mm/yyyy):
02/05/1998
Enter final date(dd/mm/yyyy):
10/05/1998

02/05/1998
03/05/1998
04/05/1998
05/05/1998
06/05/1998
07/05/1998
08/05/1998
09/05/1998
10/05/1998
Saturday,May 02,1998
Sunday,May 03,1998
Monday,May 04,1998
Tuesday,May 05,1998
Wednesday,May 06,1998
Thursday,May 07,1998
Friday,May 08,1998
Saturday,May 09,1998
Sunday,May 10,1998
ilkay@ubuntu:~/Desktop$
```

input_date.txt (~/Desktop) - ged new_date.txt (~/Desktop) - gedit

input_date.txt	new_date.txt
1 02/05/1998	1 Saturday,May 02,1998
2 03/05/1998	2 Sunday,May 03,1998
3 04/05/1998	3 Monday,May 04,1998
4 05/05/1998	4 Tuesday,May 05,1998
5 06/05/1998	5 Wednesday,May 06,1998
6 07/05/1998	6 Thursday,May 07,1998
7 08/05/1998	7 Friday,May 08,1998
8 09/05/1998	8 Saturday,May 09,1998
9 10/05/1998	9 Sunday,May 10,1998

Just as you see the program takes dates and asks again if dates are not valid. Then generates dates and finds weekday and changes format and writes to the terminal and files.