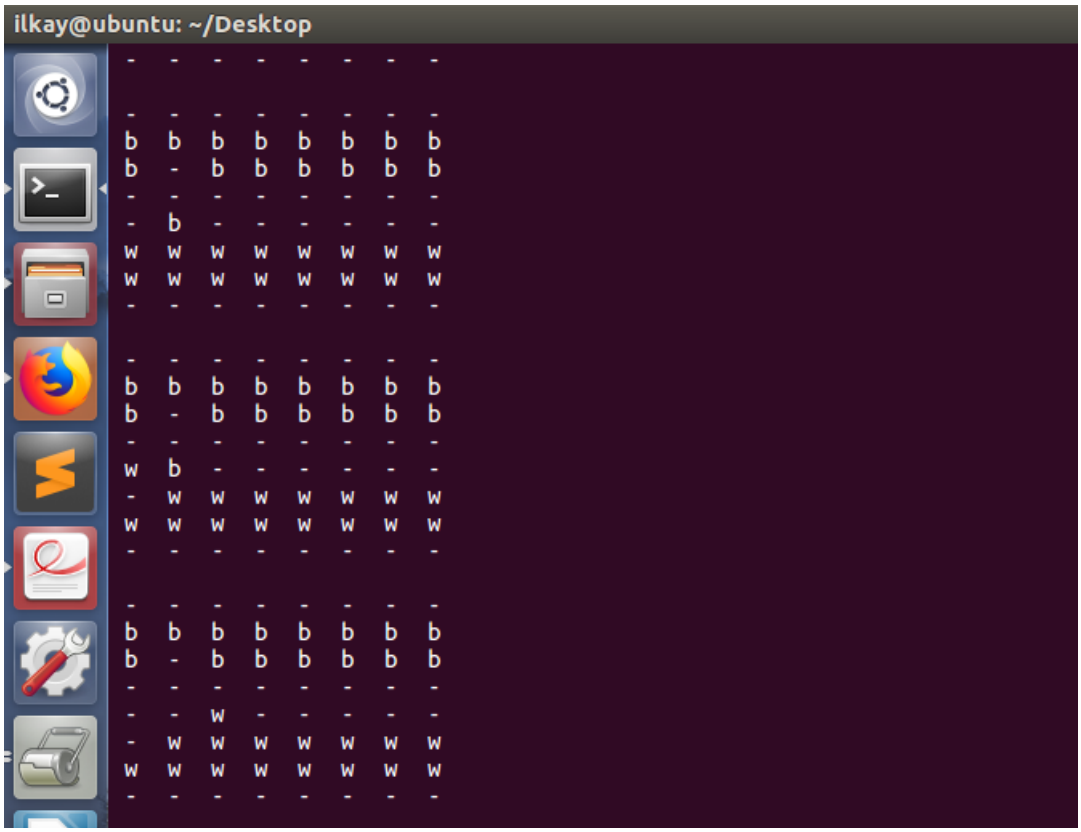
CSE102 Computer Programming HW7 Report

Part 1

In this part we were asked to write a Turkish draugths 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.

```c
void init_board(piece board[][8]){ /* board at the start*/
int i;
    for(i=0; i<8;i++){
        board[0][i]=empty;
        board[1][i]=black_man;
        board[2][i]=black_man;
        board[3][i]=empty;
        board[4][i]=empty;
        board[5][i]=white_man;
        board[6][i]=white_man;
        board[7][i]=empty;
    }

}
int check_end_of_game(piece board[][8]){ /*checks if game is over*/
piece check[64];
int a=0,whi=0, bl=0,res,i,j;
    for(i=0;i<8;i++){
        for(j=0;j<8;j++){
            check[a]=board[i][j];
            a++;
        }
    }

    for(a=0;a<64;a++){
        if(check[a]!=black_man && check[a]!=black_king){
            whi++;
        }else if(check[a]!=white_man && check[a]!=white_king){
            bl++;
        }
    }

    if(whi==64){ /*if there is no black pieces*/
        res=white;
    } else if(bl==64){ /*if there is no white pieces*/
        res=black;
    } else { /*if there are both on the board */
        res =-1;
    }
 return res;
}
void display_board(piece board[][8]){
int i,j;

    for(i=0;i<8;i++){
        for(j=0;j<8;j++){
            if(board[i][j]==empty){
                printf(" - ");
            }else if(board[i][j]==black_man){
                printf(" b ");
            }else if(board[i][j]==white_man){
                printf(" w ");
            }else if(board[i][j]==white_king){
                printf(" W ");
            }else if(board[i][j]==black_king){
                printf(" B ");
            }
        }
        printf("\n");
    }
    printf("\n");
}
```

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.



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.

```c
}
int check_date(int day, int  month, int year){ /*this function checks if date is invalid or not*/
    if(year<0 || month<1 || month>12 || day<1){
        return 0;
    }else if((month==4 || month==6 || month==9 || month == 11) && day>30){
        return 0;
    }else if((month==1 || month==3 || month==5 || month==7 || month==8 || month==10 || month == 12) && day>31){
        return 0;
    }else if(month==2 && (year%400==0) && day>29){ /*leap years*/
        return 0;
    }else if(month==2 && (year%100==0) && day>28){
            return 0;
    }else if(month==2 && (year%4==0) && day>29){
        return 0;
    }else if(month==2 && (year%4!=0) && day>28){
        return 0;
    }else {
        return 1;
    }
}
int compare_date(int day, int  month, int year, int day_f, int month_f, int year_f){ /*this function compares dates */
    if((year_f<year) || ((year_f==year)&&(month_f<month)) || ((year_f==year)&&(month_f==month)&& (day_f<day)) ){
        return 0;
    }else {
        return 1;
    }
}
int is_leap_year(int year){ /*checks if given year is a leap year*/
    if(year%400==0) {
        return 1;
    }else if(year%100==0){
            return 0;
    }else if(year%4==0){
        return 1;
    }else {
        return 0;
    }
}
int find_size(char str[]){                      /* Finds the size of a string*/
int count;
    for ( count = 0; str[count]!= 0; ++count);          /* counts until NULL*/
    return count;
}
void new_date() /*changes format of date*/{
    int day, month, year, temp, temp2=0;
```

Line 191, Column 37

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.

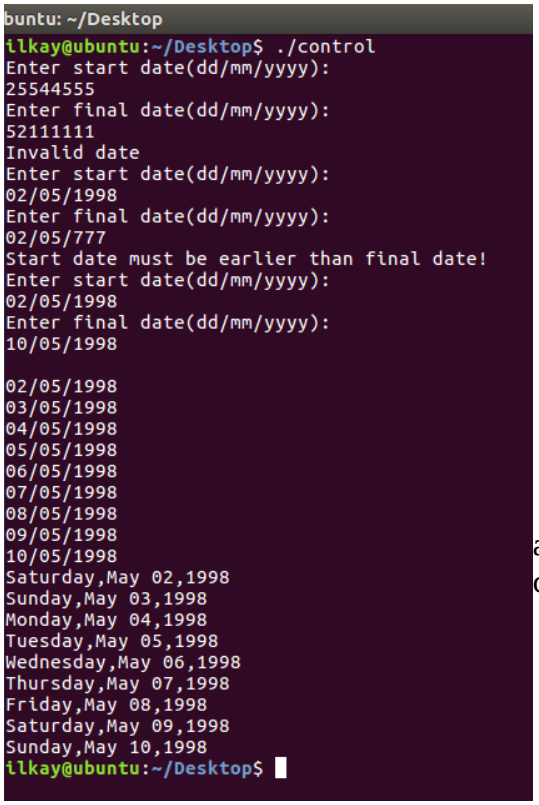
```c
void generate_days(){
int day, month, year, day_f, month_f, year_f, flag=1;
char ch;
FILE *outp;
    outp = fopen("input_date.txt", "w"); /*open file*/
    char date_i[11],date_f[11],date_n[11]; /*strings for start and final dates*/
    printf("Enter start date(dd/mm/yyyy):\n");
    scanf("%s", date_i);
    printf("Enter final date(dd/mm/yyyy):\n");
    scanf("%s", date_f);
    sscanf(date_i,"%d %c %d %c %d", &day, &ch, &month, &ch, &year);  /* turn string to integer values */
    sscanf(date_f,"%d %c %d %c %d", &day_f, &ch, &month_f, &ch, &year_f);
    if((check_date(day, month, year) && check_date(day_f, month_f, year_f))!=1){
        printf("Invalid date\n"); /* check date if its invalid ask again with recursion*/
        flag=0;
        generate_days();
    }else if(compare_date(day, month, year,day_f, month_f, year_f)!=1){ /*compare dates; first one must be earlier*/
        printf("Start date must be earlier than final date!\n");
        flag=0;
        generate_days();
    }else{
        printf("\n%s\n", date_i );
        fputs(date_i, outp);     /*write first date*/
        putc('\n', outp);
        while(flag){ /* change date until final date */
            if((month==1 || month==3 || month==5 || month==7 || month==8 || month==10 || month == 12) && day == 31){
                if(month==12){
                    day=1;
                    month=1;
                    year++;
                }else{
                    day=1;
                    month++;
                }
            }else if((month==4 || month==6 || month==9 || month == 11) && day==30){
                    day=1;
                    month++;
            }else if((month==2) && (year%4==0) && day==29) {
                    day=1;
                    month++;
            }else if((month==2) && (year%4!=0) && day==28){
                    day=1;
                    month++;
            }else {
                day++;
            }
            sprintf(date_n, "%02d%c%02d%c%02d", day,ch,month,ch,year ); /*take new date to string*/
            printf("%s\n", date_n ); /*write new date to file and terminal*/
            fputs(date_n, outp);
            putc('\n', outp);
            if(day==day_f && month==month_f && year==year_f){
                flag=0;
            }
        }
    }
    fclose(outp); /*close file*/
}
int check_date(int day, int  month, int year){ /*this function checks if date is invalid or not*/
    if(year<0 || month<1 || month>12 || day<1){
        return 0;
```

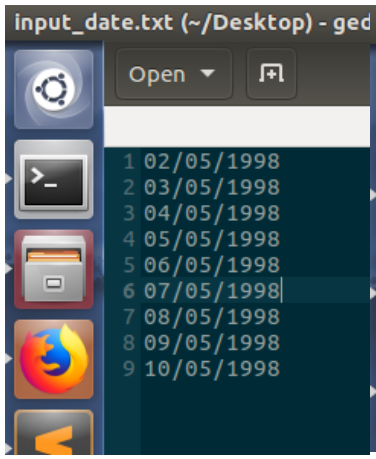
Line 191, Column 37

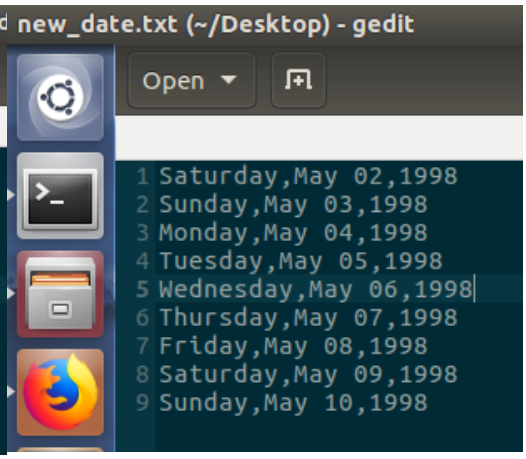
generate_days takes dates from user calls check_date and compare_dates functions to see if date are okay. Then until final date it generates dates and writes them to input_date.txt and terminal.

```c
}
void new_date() /*changes format of date*/{
    int day, month, year, temp, temp2=0;
    char ch, comma=',';
    char date[11],new_date[50];
    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*/
    int year_code[4]={4,2,0,6};
    FILE *inp;
    FILE *outp;
    inp = fopen("input_date.txt", "r");
    outp = fopen("new_date.txt", "w");
        while(fgets(date, 1024, inp)!=0){                        /*read until end of file*/
            sscanf(date,"%d %c %d %c %d", &day, &ch, &month, &ch, &year); /*turn string to int and char*/
            temp2=year; /*year may change so temp2 will hold its first value*/
            temp = (year%100)/4;
            temp+=day;
            temp+=month_key[month-1];
            if((month==1 || month==2) && (is_leap_year(year)==1)){
                temp--;
            }
            if(year>2099) { /*every 400 year repeats itself so we will subtract 400 to find year code  */
                while(year>2099){
                year = year-400;
                }
            }
            if(year>=2000){
                temp+=year_code[3];
            }else if (year>=1900){
                temp+=year_code[2];
            }else if(year>=1800){
                temp+=year_code[1];
            }else if(year>=1700){
                temp+=year_code[0];
            }
            temp+= temp2%100;
            temp=temp%7;
            if(temp==0){   /*write day to new_date string*/
                sprintf(new_date, "Saturday,");
            }else if(temp==1){
                sprintf(new_date, "Sunday,");
            }else if(temp==2){
                sprintf(new_date, "Monday,");
            }else if(temp==3){
                sprintf(new_date, "Tuesday,");
            }else if(temp==4){
                sprintf(new_date, "Wednesday,");
            }else if(temp==5){
                sprintf(new_date, "Thursday,");
            }else {
                sprintf(new_date, "Friday,");
            }

            if (month==1){ /*write month to new_date string*/
                sprintf(new_date+find_size(new_date), "January ");
            }else if(month==2){
                sprintf(new_date+find_size(new_date), "February ");
            }else if(month==3){
                sprintf(new_date+find_size(new_date), "March ");
            }else if(month==4){
                sprintf(new_date+find_size(new_date), "April ");
            }else if(month==5){
                sprintf(new_date+find_size(new_date), "May ");
            }else if(month==6){
                sprintf(new_date+find_size(new_date), "June ");
            }else if(month==7){
                sprintf(new_date+find_size(new_date), "July ");
            }else if(month==8){
                sprintf(new_date+find_size(new_date), "August ");
            }else if(month==9){
                sprintf(new_date+find_size(new_date), "September ");
            }else if(month==10){
                sprintf(new_date+find_size(new_date), "October ");
            }else if(month==11){
                sprintf(new_date+find_size(new_date), "November ");
            }else if(month==12){
                sprintf(new_date+find_size(new_date), "December ");
            }
            sprintf(new_date+find_size(new_date),"%02d%c%02d", day, comma, temp2); /*add day of month and year to string*/
            fputs(new_date, outp);
            putc('\n', outp);                                    /*write to the file*/
            printf("%s\n" , new_date);                               /*printf for terminal*/
        }
    fclose(inp);
    fclose(outp);                                                /*close files*/
}
```

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.







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.