Problem solving steps

Problem Analysis:

The main objective of our program needed to be simplicity and quick access. For this, we all did a bit of brainstorming and went for the task.

The main sections that needed emphasis were:

1. Use of a nominal graphical interface.
2. Use of file handling techniques.
3. Use of English to Nepali Date Converter.

Since we were not using the 16-bit Turbo C compiler to develop this program, we needed to create our own graphics library. We needed graphics for drawing boxes, use of colors in text and background.

Next important part was the Date converter, as in Nepal we use the Bikram Sambat Time instead of the AD that the windows system uses. So we needed to create our own date converter for the cause.

As we were to allow the user to create his own loadshedding schedule, we needed to use file handling techniques like opening, modifying and saving of files.

We also needed to create unsupported functions for windows like **gotoxy()**, textcolor(), cprintf(), drawbox(),etc.

The main analysis included knowing how the Nepal Electricity Authority(NEA) circulates its loadshedding pattern. For it, we needed to know the no. of loadshedding groups and their respective power supply timings. This schedule was set as default schedule in our program, i.e when the user views the schedule directly without any modification, he will be able to see the one set according to the NEA. For it, we needed to make use of a default file which contained the information.

Later when the user makes any modification to the schedule, a new file is created containing the latest modification information, so that this modified file is opened when the loadshedding schedule is viewed.

There's also a Power Control Menu. With it, we aim to let the user force change the status of any specified loadshedding group. For it, suppose I'm in loadshedding group -1, and the timings suggest that I have a power cut in normal. In this case, the program shows red mark in my group for the time the application is opened. But if I want to force change the program so that power is on (works for the changed schedule) at the time, I'll go to the Power Control Menu, enter Power On, specify the group and I'm done. To make this possible , we had to revert the conditions that controlled the status of power supply.

Algorithm:

Step 1: Start

Step 2: Run main_menu () block

Step 3: If user selects Make LS schedule, run make_ls_schedule () block

Step 4: If user selects View LS schedule, run show_now_ls () block

Step 5: If user selects Power Control, run power_control () block

Step 6: Else exit

Step 7: Stop

Note: It must not be considered that the above algorithm represents the entire program's working steps. The above algorithm only refers to the steps of the main function. Other details containing the steps for other functions are their definitions in header files are given below.

Details of the Program:

Full descriptions of the header files and functions used are given below:

1. **The main() function:**

```
int main()
{  ShowConsoleCursor(NO);
   TextColor(240);
   system("cls");
   TextColor(112);
   DrawBox2(0,1,80,23,177);
   TXTprintf("ASCS",15,6,121,0);
   TextColor(55);
   DrawBox0(14,16,44,5,32);
   LoadingAnimation(20,17,30,10,"Loading...",55);
   Sleep(100);
   TextColor(112);
   DrawBox0(14,16,44,5,177);
   main_menu();
   return 0;}
```

Most of the syntax in the function are self descriptive. It is the foremost step run in the program. Firstly the background is printed in the box drawn using the function Drawbox2(). Then inside the box, "ASCS" text is printed which uses the TXTprintf() function. Below it, another box(using function DrawBox0()) is printed for the Loading… animation. For the animation, LoadingAnimation() function is used. Finally the screen is made blank with a empty box drawn using Drawbox0(). After it, the main() function calls the main_menu() function which will be dealt with later.

Note:

Drawbox0(): Draw box with no border

Drawbox2(): Draw box with double borders

2. The main_menu() function:

```
void main_menu()
{   char chs;
    TextColor(240);
    system("cls");
    int menu_s;
    xnMenu main_xnMenu;
    ShowConsoleCursor(NO);
    TextColor(112);
    DrawBox0(0,1,80,23,177);
    TXTprintf("ASCS",19,6,121,0);
    DrawHorizontalLine2(19,16,42);
    gotoxy(27,16);  printf("AutoShedding Control System.");
    gotoxy(3,0);    cprintf("Main Menu",128);
    TextColor(240);
    gotoxy(1,0);    printf("%c",240);
    gotoxy(1,24);   printf("Select One Option.");
    TextColor(128);
    main_xnMenu.lx=2;main_xnMenu.ly=1,main_xnMenu.items=5;main_xnMenu.width=30;main_xnMenu.now_pos=1;
    main_xnMenu.item_char[1]="Make LS schedule";
    main_xnMenu.item_char[2]="View LS schedule";
    main_xnMenu.item_char[3]="Power Control";
    main_xnMenu.item_char[4]="-";
    main_xnMenu.item_char[5]="Exit";
    GO_TOP
    while(!kbhit()){
          TextColor(240);
          gotoxy(52,0);   print_now_nepali_date();
          gotoxy(71,24);  print_now_nepali_time();
          Sleep(10);
    }
    chs=getch();
    if(chs=='m' || chs=='M' || chs==78){
       print_xnMenu(main_xnMenu);
       menu_s=handle_xnMenu(main_xnMenu);
    switch(menu_s){
    case 1:  make_ls_schedule();  break;
    case 2: show_now_loadshedding();  main_menu(); break;
    case 3: power_control(); break;
    case 5: DrawBox2(0,20,80,4,32); gotoxy(2,21); printf("Are you sure to exit?(\"system may crash\")(y/n):");  if(getchar()=='y' ||
getchar()=='Y')      exit(EXIT_SUCCESS);     else     main_menu();   break;    default:  main_menu();  break; } }     else main_menu();  }
```

In this block as well, first “ASCS” is printed in the background same as in the main() function. The additional to this function is that it presents a menu at the left top corner of the console screen. For it, **gotoxy()** function is used. Here we’ve used the contents of a structure main_xnMenu(which is defined in the header xnMenu.h). The contents are assigned into main_xnMenu.item_char[]. Here, we assign the values as:

1. main_xnMenu.item_char[1]="Make LS schedule";
2. main_xnMenu.item_char[2]="View LS schedule";
3. main_xnMenu.item_char[3]="Power Control";
4. main_xnMenu.item_char[4]="-";
5. main_xnMenu.item_char[5]="Exit";

This assignment is used below in a **switch-case** ladder. This ladder uses the paramaters defined above to call the functions as:

1. - make_ls_schedule()
2. - show_now_loadshedding() and main_menu()
3. - power_control()
5. - exit if user agrees
   Else call back the main_menu() function

(The default case is calling the main_menu() function)

The functions 1,2 and 3 are to be dealt with below.

The make_ls_schedule() function:

```
void make_ls_schedule()

{

    int ms,i;
    char file_name[20]=LS_SCHEDULE_FILE_NAME;
    ls_group grp;
    Time td,default_morning_start={1,0,0},default_night_start={14,0,0};
    gotoxy(1,24);   printf("Choose one option.");
    xnMenu mnu_mls;
    mnu_mls.lx=25; mnu_mls.ly=2;mnu_mls.width=25;mnu_mls.items=3; mnu_mls.now_pos=1; mnu_mls.last_pos=1;
      mnu_mls.item_char[1]="Time Duration.";
      mnu_mls.item_char[2]="One day time";
      mnu_mls.item_char[3]="All week time";
    print_xnMenu(mnu_mls);
    ms=handle_xnMenu(mnu_mls);
      switch(ms){
        case 1:
          TextColor(112);
            system("cls");
            DrawBox1(0,1,80,23,32);
            gotoxy(1,24);   printf("This will make loadshedding schedule using default time.");
            gotoxy(1,3);    printf("Enter Duration of loadshedding(hh/mm/ss):");
            ShowConsoleCursor(YES);
            scanf("%d%d%d",&td.hour,&td.minute,&td.second);
            ShowConsoleCursor(NO);
            auto_schedule_make(file_name,td,default_morning_start,default_night_start);
            Sleep(200);
            gotoxy(1,24); printf("Schedule made successfuflly..                                                    ");
            Sleep(1000);
            main_menu();
        break;
        case 2:
          TextColor(112);
            system("cls");
            DrawBox1(0,1,80,23,32);
            gotoxy(1,24);   printf("This will make loadshedding schedule using time you provide.");
            gotoxy(1,3);    printf("Enter Duration of loadshedding(hh/mm/ss):");
            ShowConsoleCursor(YES);
            scanf("%d%d%d",&td.hour,&td.minute,&td.second);
            gotoxy(1,4);
            printf("Enter least morning time to start loadshedding:");

    scanf("%d%d%d",&default_morning_start.hour,&default_morning_start.minute,&default_morning_start.second);
            gotoxy(1,5);
            printf("Enter least night time to start loadshedding:");
            scanf("%d%d%d",&default_night_start.hour,&default_night_start.minute,&default_night_start.second);
            ShowConsoleCursor(NO);
            auto_schedule_make(file_name,td,default_morning_start ,default_night_start);
            Sleep(200);
```

```
            gotoxy(1,24); printf("Schedule made successfullly..");


  Sleep(1000);
            main_menu();
       break;
case 3:
          TextColor(112);
            system("cls");
            gotoxy(1,24);   printf("This will make loadshedding schedule using group-1 time.");
          for(i=1;i<=7;i++){
            DrawBox1(0,1,80,23,32);
            gotoxy(1,3);
            ShowConsoleCursor(YES);
            printf("Enter loadshedding time morning of day %d(hh/mm/ss):",i);

scanf("%d%d%d",&grp.day[i].morning[0].hour,&grp.day[i].morning[0].minute,&grp.day[i].morning[0].second);
            gotoxy(1,4);    printf("to (hh/mm/ss):");

scanf("%d%d%d",&grp.day[i].morning[1].hour,&grp.day[i].morning[1].minute,&grp.day[i].morning[1].second);
            gotoxy(1,5);    printf("Enter loadshedding time night of day %d(hh/mm/ss):",i);
            scanf("%d%d%d",&grp.day[i].night[0].hour,&grp.day[i].night[0].minute,&grp.day[i].night[0].second);
            gotoxy(1,6);    printf("to (hh/mm/ss):");
            scanf("%d%d%d",&grp.day[i].night[1].hour,&grp.day[i].night[1].minute,&grp.day[i].night[1].second);
          }
          auto_schedule_make_week_time(file_name,grp);
          Sleep(200);
            gotoxy(1,24); printf("Schedule made successfullly..");
            Sleep(1000);
          main_menu();
       break;
       default:
          main_menu();
       break;
       }
    }
```

As the name suggests, this is a function for generating a loadshedding schedule according to the input of time supplied by the user. For it, the user is given three choices:

1. Provide time duration for a day(and the schedule for the whole week will be automatically generated.)
2. Provide all the timings for a day and the schedule for a week will be automatically generated.
3. Provide all the timings for the whole week and get a schedule for the week.

Note:

Case1 will use make the schedule using the default time set already.

Case2 will make the schedule using the least morning and night time provided by the user.

Case3 will take all the input of a week for a group and use it to make the schedule for all other groups.

In this process also, another function not defined in the main() function, i.e auto_schedule_make() is used to generate the loadshedding schedule. This function is defined in another header **Autoshedding.h** which is to be introduced later.

Other than this, everything is self-descriptive and can be understood well when analysed thoroughly.

4. **The show_now_loadshedding() function:**

   This function prints the loadshedding created in the make_ls_schedule() function. It has been defined under **show_now_ls.h** header as:

```
int show_now_loadshedding()
{
Tab grp_tab;
int a;
for(a=1;a<=7;a++)
   group[a]=load_group_ls_time(LS_SCHEDULE_FILE_NAME,a);
ShowConsoleCursor(0);
grp_tab.lx=0;
grp_tab.ly=1;
grp_tab.height=22;
grp_tab.width=80;
grp_tab.items=7;
grp_tab.now_pos=1;
grp_tab.item_char[1]="Group-01";
grp_tab.item_char[2]="Group-02";
grp_tab.item_char[3]="Group-03";
grp_tab.item_char[4]="Group-04";
grp_tab.item_char[5]="Group-05";
grp_tab.item_char[6]="Group-06";
grp_tab.item_char[7]="Group-07";
TextColor(112);
DrawBox0(0,0,80,1,TAB_BG_TEXT);   gotoxy(1,0); putchar(240);
gotoxy(3,0);    printf("Loadshedding Schedule",254);
draw_tab(grp_tab);
TextColor(151);   DrawBox0(0,23,80,1,TAB_BG_TEXT);
TextColor(112);   DrawBox0(0,24,80,1,TAB_BG_TEXT);
gotoxy(0,24);
cprintf("\x1B",112); cprintf("ESC   R",115); cprintf("efresh     ",112);
gotoxy(0,0);
handle_tab_control(grp_tab);
getch();
}
```

This function is deployed to read the contents of the file saved after creating the data supplied by the user. This task is done by the load_group_ls_time() function. Here, a graphical interface is shown on the screen using boxes and colored texts.

Header files :

1. Autoshedding.h

```
#ifndef AUTOSHEDDING_H_INCLUDED
#define AUTOSHEDDING_H_INCLUDED

  #include<stdio.h>
  #include"ls.h"
 ///Function to make a loadshedding schedule and print it in a file
  void auto_schedule_make(char file_name[32],Time time_a_day,Time morning_start_time,Time night_start_time)
  {
  ///define variables
     FILE *file_ls_schedule;
     int time_a_day_seconds,time_morning_seconds,time_night_seconds,i,j;
     Time morning_time,night_time;   ///times structure to store loadshedding time of morning and night.
  ///Initialization
     file_ls_schedule=fopen(file_name,"w");
     grps[1].day[1].morning[0]=morning_start_time;
     grps[1].day[1].night[0]=night_start_time;
     time_a_day_seconds=(time_a_day.hour*3600+time_a_day.minute*60+time_a_day.second);///store time in second.
     time_morning_seconds=(time_a_day_seconds/2);      ///divide time by 2 for morning..
     morning_time.hour=(time_morning_seconds/3600);
     morning_time.minute=(time_morning_seconds-morning_time.hour*3600)/60;
     morning_time.second=(time_morning_seconds-morning_time.hour*3600-morning_time.minute*60);
     time_night_seconds=(time_a_day_seconds-time_morning_seconds);
     night_time.hour=(time_night_seconds/3600);
     night_time.minute=(time_night_seconds-night_time.hour*3600)/60;
     night_time.second=(time_night_seconds-night_time.hour*3600-night_time.minute*60);
     grps[1].day[1].morning[1]=add_time(grps[1].day[1].morning[0],morning_time);
     grps[1].day[1].night[1]=add_time(grps[1].day[1].night[0],night_time);
     grps[1]=initialize_group_ls_time(grps[1],1);
     for(i=2;i<=7;i++)
        grps[i]=copy_group_times(grps[1],(i-1));
fprintf(file_ls_schedule,"\t\t\t\t\tSUN.\t\t\t\t\t\tMON.\t\t\t\t\t\tTUE.\t\t\t\t\t\tWED.\t\t\t\t\t\tTHU.\t\t\t\t\t\tFRI.\t\t\t\t\t
\tSAT.");
     for(i=1;i<=7;i++){
     fprintf(file_ls_schedule,"\nGroup-%d\t\t\t",i);
        for(j=1;j<=7;j++){
        time_fprintf(file_ls_schedule,grps[i].day[j].morning[0]);   fprintf(file_ls_schedule,"-");
        time_fprintf(file_ls_schedule,grps[i].day[j].morning[1]);   fprintf(file_ls_schedule,"\t\t\t");
        }
        fprintf(file_ls_schedule,"\n\t\t\t\t");
        for(j=1;j<=7;j++){
        time_fprintf(file_ls_schedule,grps[i].day[j].night[0]);    fprintf(file_ls_schedule,"-");
        time_fprintf(file_ls_schedule,grps[i].day[j].night[1]);    fprintf(file_ls_schedule,"\t\t\t");
        }
        fprintf(file_ls_schedule,"\n");
     }
  fclose(file_ls_schedule);
```

```
   }

   void auto_schedule_make_week_time(char file_name[16],ls_group grp)
   {
      FILE *file_ls_schedule;
      int i,j;
      file_ls_schedule=fopen(file_name,"w");
      for(i=1;i<=7;i++)
         grps[i]=copy_group_times(grp,(i-1));

fprintf(file_ls_schedule,"\t\t\t\t\tSUN.\t\t\t\t\t\tMON.\t\t\t\t\t\tTUE.\t\t\t\t\t\tWED.\t\t\t\t\t\tTHU.\t\t\t\t\t\tFRI.\t\t\t\t
\t\tSAT.");
      for(i=1;i<=7;i++){
      fprintf(file_ls_schedule,"\nGroup-%d\t\t\t",i);
         for(j=1;j<=7;j++){
         time_fprintf(file_ls_schedule,grps[i].day[j].morning[0]);   fprintf(file_ls_schedule,"-");
         time_fprintf(file_ls_schedule,grps[i].day[j].morning[1]);   fprintf(file_ls_schedule,"\t\t\t");
         }
         fprintf(file_ls_schedule,"\n\t\t\t\t");
         for(j=1;j<=7;j++){
         time_fprintf(file_ls_schedule,grps[i].day[j].night[0]);    fprintf(file_ls_schedule,"-");
         time_fprintf(file_ls_schedule,grps[i].day[j].night[1]);    fprintf(file_ls_schedule,"\t\t\t");
         }
         fprintf(file_ls_schedule,"\n");
      }
   fclose(file_ls_schedule);
}
#endif // AUTOSHEDDING_H_INCLUDED
```

As the source-code shows, this header has two main functions:

a. void auto_schedule_make():
   This function makes a loadshedding schedule using the default morning and night time and prints the time for the whole week it in a file. Under it, the input time is converted into the 12 hour format, and printed to the file.

b. void auto_schedule_make_week_time():
   This function makes loadshedding schedule using the time provided by the user for the whole week and prints it to the file.

2. ls.h

```
#ifndef LS_H
#define LS_H
   #define MAX_PLACES_IN_A_GROUP 32
   #define NUMBER_OF_GROUPS 7
   #include<stdio.h>
   #include<windows.h>
   #include<time.h>
   #define LS_SCHEDULE_FILE_NAME "ls-schedule.ls"

 ///   #include "D:\c\GR_H\DateConverter.h"


   typedef struct{
            unsigned int hour,minute,second;
   }Time;
   typedef struct{
            Time morning[2],day[2],night[2];
   }Day;
   typedef struct{
            unsigned int number;
            Day day[8];///0,1-morning time,2,3-day time,4,5-evening time; day[1] is starting and is
indicated to sunday.
            int places_id[MAX_PLACES_IN_A_GROUP];
            int state,is_off; ///shows wheather perticular group is on or off 1-ON 0-OFF
   }ls_group;

   ls_group grps[(NUMBER_OF_GROUPS+1)];


   Time copy_time(Time source)
   {
      Time dest;
      dest.hour=source.hour;
      dest.minute=source.minute;
      dest.second=source.second;
   return dest;
   }
   ls_group copy_group_times(ls_group g_source,int diff_day) /* diff_day --> difference of day between
two group g1 and g2 having same schedule*/
   {
      ls_group g_dist;
      int i,i1;
      for(i=1;i<=7;i++){
            if((i+diff_day)<=7)
              i1=(i+diff_day);
            else
```

```
            i1=(i+diff_day-7);
        g_dist.day[i1].morning[0]=(g_source.day[i].morning[0]);
        g_dist.day[i1].morning[1]=(g_source.day[i].morning[1]);
        g_dist.day[i1].night[0]=(g_source.day[i].night[0]);
        g_dist.day[i1].night[1]=(g_source.day[i].night[1]);
      }
   return g_dist;
   }
   Time add_time(Time t1,Time t2)
   {
      Time sum;
      sum.hour=t1.hour+t2.hour;
      sum.minute=t1.minute+t2.minute;
      sum.second=t1.second+t2.second;
      if (sum.second>=60){
         ++sum.minute;
         sum.second-=60;
      }
      if(sum.minute>=60){
         ++sum.hour;
         sum.minute-=60;
      }
      return sum;
   }
   Time subtract_time(Time t1,Time t2)
   {
      Time sub;
      int seconds;
      if((t1.hour>t2.hour) || (t1.hour==t2.hour && t1.minute>t2.minute) || (t1.hour==t2.hour &&
t1.minute==t2.minute && t1.second>t2.second)){
        seconds=(t1.hour*3600+t1.minute*60+t1.second)-(t2.hour*3600+t2.minute*60+t2.second);
        sub.hour=(seconds/3600);
        sub.minute=(seconds-sub.hour*3600)/60;
        sub.second=(seconds-sub.hour*3600-sub.minute*60);
      }
      else{
         seconds=(t2.hour*3600+t2.minute*60+t2.second)-(t1.hour*3600+t1.minute*60+t1.second);
        sub.hour=(seconds/3600);
        sub.minute=(seconds-sub.hour*3600)/60;
        sub.second=(seconds-sub.hour*3600-sub.minute*60);
      }
   return sub;
   }
```

```
      ///Increament function
Time time_increase(Timesource_actual,Time increase_time)
{
   Time source,time_temp;
   source=copy_time(source_actual);
         time_temp=add_time(source,increase_time);
   source=copy_time(time_temp);
return source;
}

void time_printf(Time t1)
{
printf("%.2u:%.2u",t1.hour,t1.minute);
}
void time_fprintf(FILE *fle,Time t1)
{
fprintf(fle,"%.2u:%.2u:%.2u",t1.hour,t1.minute,t1.second);
}
void file_printf(char file_name[32])
{
   FILE *ls_file;
   char ch_file;
   ls_file=fopen(file_name,"r");
   if(ls_file==NULL)
      printf("\n\tNO loadshedding Schedule Found!\n\tFirst create a loadshedding schedule...");
   else{
      while((ch_file=fgetc(ls_file))!=EOF)
         printf("%c",ch_file);
   }
fclose(ls_file);
}

///Function to take the default time on making loadshedding schedule
Time use_default_ls_time(char   ver) ///character to verify which time is to make default 'm'-> morning
{
   Time t1;
   if(ver=='m'){
      t1.hour=3;
      t1.minute=0;
      t1.second=0;
   }
   else if(ver=='d'){
      t1.hour=12;
      t1.minute=0;
      t1.second=0;
```

```
        }
        else if(ver=='n'){
          t1.hour=13;
          t1.minute=0;
          t1.second=0;
        }
        else{
          t1.hour=0;
          t1.minute=0;
          t1.second=0;
        }
    return t1;
    }
    ///Function to initialize all the time of a week one day's time given..
/**Declearation of time***/Time time_increase_duration={1,0,0};
    ls_group initialize_group_ls_time(ls_group g1,int day)
    {
        int i;
        ls_group g2=g1;
        for(i=1;i<=7;i++){
          if(i==day)
            continue;
            ///Increase or decrease time with ls time.
          else if(i>day){
            g2.day[day].morning[0]=(add_time(g2.day[day].morning[0],time_increase_duration));
            g2.day[day].morning[1]=(add_time(g2.day[day].morning[1],time_increase_duration));
            g2.day[day].night[0]=(add_time(g2.day[day].night[0],time_increase_duration));
            g2.day[day].night[1]=(add_time(g2.day[day].night[1],time_increase_duration));
          }
          else if(i<day){
            g2.day[day].morning[0]=(subtract_time(g2.day[day].morning[0],time_increase_duration));
            g2.day[day].morning[1]=(subtract_time(g2.day[day].morning[1],time_increase_duration));
            g2.day[day].night[0]=(subtract_time(g2.day[day].night[0],time_increase_duration));
            g2.day[day].night[1]=(subtract_time(g2.day[day].night[1],time_increase_duration));
          }
          g1.day[i].morning[0]=(g2.day[day].morning[0]);
          g1.day[i].morning[1]=(g2.day[day].morning[1]);
          g1.day[i].night[0]= (g2.day[day].night[0]);
          g1.day[i].night[1]=(g2.day[day].night[1]);
        }
    return g1;
    }

void fprintf_ls_schedule(char file_name[32],ls_group *grps)
{
FILE *file_ls_schedule;
int i,j;
    file_ls_schedule=fopen(file_name,"w");
fprintf(file_ls_schedule,"\t\t\t\t\tSUN.\t\t\t\t\t\tMON.\t\t\t\t\t\tTUE.\t\t\t\t\t\t
WED.\t\t\t\t\t\tTHU.\t\t\t\t\t\tFRI.\t\t\t\t\t\tSAT.");
        for(i=1;i<=7;i++){
```

```
        fprintf(file_ls_schedule,"\nGroup-%d\t\t\t",i);
          for(j=1;j<=7;j++){
          time_fprintf(file_ls_schedule,grps[i].day[j].morning[0]);   fprintf(file_ls_schedule,"-");
          time_fprintf(file_ls_schedule,grps[i].day[j].morning[1]);   fprintf(file_ls_schedule,"\t\t\t");
          }
          fprintf(file_ls_schedule,"\n\t\t\t\t");
          for(j=1;j<=7;j++){
          time_fprintf(file_ls_schedule,grps[i].day[j].night[0]);    fprintf(file_ls_schedule,"-");
          time_fprintf(file_ls_schedule,grps[i].day[j].night[1]);    fprintf(file_ls_schedule,"\t\t\t");
          }
          fprintf(file_ls_schedule,"\n");
        }
    fclose(file_ls_schedule);
}
Time get_now_time()
{
Time tme;
SYSTEMTIME systime;
    GetLocalTime(&systime);
    tme.hour=systime.wHour;
    tme.minute=systime.wMinute;
    tme.second=systime.wSecond;
return tme;
}
///Function to load loadshedding database from .ls file saved by the program..
Time load_ls_time(FILE *fl)
{
Time ls_tm={0,0,0};;
    fscanf(fl,"%d",&ls_tm.hour);
    fseek(fl,1,SEEK_CUR);
    fscanf(fl,"%d",&ls_tm.minute);
    fseek(fl,1,SEEK_CUR);
    fscanf(fl,"%d",&ls_tm.second);
return ls_tm;
}
ls_group load_group_ls_time(char f_name[32],unsigned int grp_no)
{
int i,cursor_pos_1=81,cursor_pos_2=226;
ls_group grps[3];
FILE *ls_schedule;
ls_schedule=fopen(f_name,"r");
fseek(ls_schedule,cursor_pos_1,SEEK_SET);
    for(i=1;i<=7;i++){
        grps[1].day[i].morning[0]=load_ls_time(ls_schedule);
        fseek(ls_schedule,1,SEEK_CUR);
        grps[1].day[i].morning[1]=load_ls_time(ls_schedule);
        fseek(ls_schedule,3,SEEK_CUR);
    }
    fseek(ls_schedule,cursor_pos_2,SEEK_SET);
    for(i=1;i<=7;i++){
        grps[1].day[i].night[0]=load_ls_time(ls_schedule);
```

```
        fseek(ls_schedule,1,SEEK_CUR);
        grps[1].day[i].night[1]=load_ls_time(ls_schedule);
        fseek(ls_schedule,3,SEEK_CUR);
    }
        grps[2]=copy_group_times(grps[1],(grp_no-1));
        return grps[2];
  fclose(ls_schedule);
  }
  int is_greater_time(Time t1,Time t2)
  {
    if(t1.hour>t2.hour || (t1.hour==t2.hour && t1.minute>t2.minute) || (t1.hour==t2.hour &&
  t1.minute==t2.minute && t1.second>t2.second))
        return 1;
    else
        return 0;
  }
  #endif // LS_H_INCLUDED
```

At the start of the header, three structures Time, Day and ls_group are initialized. Then the windows function The main functions used in this header are described below:

a. Time copy_time() – copies the time supplied
b. copy_group_times() –
c. Time add_time() –
d. Time subtract_time() –
e. Time time_increase() –
f. file_printf() – prints the loadshedding file using file
g. Time use_default_ls_time() – takes the default time on making the loadshedding schedule
h. initialize_group_ls_time() – initializes all the time of a week when one day's time is given
i. Time get_now_time() – gets the system time
j. Time load_ls_time() – loads the loadshedding database from the .ls file saved by the program

3. show_now_ls.h

```
#ifndef SHOW_NOW_LS_H
#define SHOW_NOW_LS_H
#include<stdio.h>
#include"graphics_lib.h"
#include"ls.h"
#include"DateConverter.h"
#define TAB_BG_TEXT 32
#define NO 0
#define YES !NO
#define ON 1
#define OFF !ON
ls_group group[8];
   typedef struct{
            int lx,ly,width,height,items,now_pos;
            char *item_char[32];
   }Tab;
void refresh_group_state(ls_group *grp)
{
Time now,ls_morning[2],ls_night[2];
Date now_date=get_now_bs_date();
   now=get_now_time();
   ls_morning[0]=grp->day[now_date.day+1].morning[0];
   ls_morning[1]=grp->day[now_date.day+1].morning[1];
   ls_night[0]=grp->day[now_date.day+1].night[0];
   ls_night[1]=grp->day[now_date.day+1].night[1];
if((is_greater_time(ls_morning[1],now) && is_greater_time(now,ls_morning[0])) ||
(is_greater_time(ls_night[1],now) && is_greater_time(now,ls_night[1])))
   grp->state=ON;
else
   grp->state=OFF;
}
void draw_ls_schedule_tab_content(ls_group grp)
{
Time time_went;
int i,j,min_v=1,max_v=4,a;
   grp.is_off=NO;
   refresh_group_state(&grp);
   for(j=0;j<=1;j++){
   for(i=min_v;i<=max_v;i++){
      ///DrawBox0(4+j*38,i*4-j*4*4+(i-min_v),35,4);
      TextColor(126);
      gotoxy(5+j*37,i*4-j*4*4+(i-min_v)-1);
      switch(i){
         case 1:
            printf("Aaitabar.");
            break;
         case 2:
            printf("Sombar.");
            break;
         case 3:
            printf("Mangalbar.");
            break;
         case 4:
            printf("Budhabar.");
            break;
```

```
        case 5:
           printf("Bihibar.");
           break;
        case 6:
           printf("Sukrabar.");
           break;
        case 7:
           printf("Sanibar.");
           break;
        }
        if(which_day_BS(get_now_bs_date())==(i-1)){
           gotoxy(42,18);  printf("Status %c",28);
           if(grp.state==ON && grp.is_off==NO)
              TextColor(112);
           else
              TextColor(124);
           gotoxy(48,18);
           printf("%c",220);
           gotoxy(48,19);
           printf("%c",223);
              gotoxy(49,18);
           printf("%c",220);
           gotoxy(49,19);
           printf("%c",223);
           TextColor(116);
        }else
           TextColor(113);
     gotoxy(5+j*37,i*4-j*4*4+(i-min_v));    printf("Morning:\t%.2d:%.2d:%.2d -
%.2d:%.2d:%.2d",grp.day[i].morning[0].hour,grp.day[i].morning[0].minute,grp.day[i].morning[0].second,grp
.day[i].morning[1].hour,grp.day[i].morning[1].minute,grp.day[i].morning[1].second);
     gotoxy(5+j*37,1+i*4-j*4*4+(i-min_v));    printf("Nigth:\t%.2d:%.2d:%.2d -
%.2d:%.2d:%.2d",grp.day[i].night[0].hour,grp.day[i].night[0].minute,grp.day[i].night[0].second,grp.day[i].nig
ht[1].hour,grp.day[i].night[1].minute,grp.day[i].night[1].second);
     gotoxy(5+j*37,2+i*4-j*4*4+(i-min_v));
     TextColor(127);

  }
  min_v=max_v+1;
  max_v=7;
}
}
void draw_tab(Tab tb)
{
  int i,j,min_v=1,max_v=4;
  TextColor(241);
  DrawBox1(tb.lx,tb.ly+1,tb.width,tb.height-1,TAB_BG_TEXT);
  gotoxy(tb.lx,tb.ly);    cprintf(" ",144);
  for(i=1;i<=tb.items;i++){
     if(i==tb.now_pos)
          TextColor(240);
     else
        TextColor(144);
     gotoxy(tb.lx+1+(i-1)*11,tb.ly);   printf(" %s  ",tb.item_char[i]);
  TextColor(144);
  }
  gotoxy(tb.lx+tb.width,tb.ly+tb.height);   cprintf("  ",144);
  for(j=0;j<=1;j++){
```

```
      for(i=min_v;i<=max_v;i++){
      TextColor(126);
      DrawBox0(4+j*37,i*4-j*4*4+(i-min_v)-1,35,4,TAB_BG_TEXT);
      }
   min_v=max_v+1;
   max_v=8;
   }
   draw_ls_schedule_tab_content(group[tb.now_pos]);
}
void refresh_draw_tab(Tab tb,int last_now_pos)
{
      TextColor(144);
      gotoxy(tb.lx+1+(last_now_pos-1)*11,tb.ly);   printf(" %s  ",tb.item_char[last_now_pos]);
      TextColor(240);
      gotoxy(tb.lx+1+(tb.now_pos-1)*11,tb.ly);   printf(" %s  ",tb.item_char[tb.now_pos]);
}

void handle_tab_control(Tab tb)
{
char ch;
int in,day,a;
Time t_gone,t_remaining,now,morning[2],night[2];
   while(!kbhit()){
   day=which_day_BS(get_now_bs_date())+1;
   now=get_now_time();
   morning[0]=group[tb.now_pos].day[day].morning[0];
   morning[1]=group[tb.now_pos].day[day].morning[1];
   night[0]=group[tb.now_pos].day[day].night[0];
   night[1]=group[tb.now_pos].day[day].night[1];
   if(group[tb.now_pos].state==YES){
      if(is_greater_time(morning[1],now) && is_greater_time(now,morning[0])){
         t_remaining=subtract_time(morning[1],now);
         t_gone=subtract_time(now,morning[0]);
      }else if(is_greater_time(now,night[1])){
         morning[0]=group[tb.now_pos].day[day+1].morning[0];
         morning[0].hour+=24;
         t_remaining=subtract_time(morning[0],now);
         t_gone=subtract_time(now,night[1]);
      }
      else{
         t_remaining=subtract_time(night[1],now);
         t_gone=subtract_time(now,night[0]);
      }
   }else{
      if(is_greater_time(now,morning[1]) && is_greater_time(night[0],now)){
         t_remaining=subtract_time(night[0],now);
         t_gone=subtract_time(now,morning[1]);
      }else{
         morning[0].hour+=24;
         t_remaining=subtract_time(morning[0],now);
         t_gone=subtract_time(now,night[1]);
      }
   }
      TextColor(112);
      gotoxy(50,24);  print_now_nepali_date();
      gotoxy(70,0);
printf("%.2d:%.2d:%.2d",get_now_time().hour,get_now_time().minute,get_now_time().second);
```

```
    gotoxy(42,20);
    TextColor(127);
    printf("(");
    time_printf(t_gone);    printf(",- "); time_printf(t_remaining);
    printf(")");
  Sleep(0);
    }
  ch=getch();
  switch(ch){
    case 75:    ///left key
      if(tb.now_pos>1){
        tb.now_pos--;
        refresh_draw_tab(tb,(tb.now_pos+1));
        draw_ls_schedule_tab_content(group[tb.now_pos]);
      }
      else if(tb.now_pos==1){
        tb.now_pos=tb.items;
        refresh_draw_tab(tb,1);
        draw_ls_schedule_tab_content(group[tb.now_pos]);
      }
      handle_tab_control(tb);
      break;
    case 77:    ///right key
      if(tb.now_pos<tb.items){
          tb.now_pos++;
        refresh_draw_tab(tb,(tb.now_pos-1));
        draw_ls_schedule_tab_content(group[tb.now_pos]);
      }
      else if(tb.now_pos==tb.items){
        tb.now_pos=1;
        refresh_draw_tab(tb,tb.items);
        draw_ls_schedule_tab_content(group[tb.now_pos]);
      }
      handle_tab_control(tb);
      break;
    case 49:
      in=tb.now_pos;
      tb.now_pos=1;
      refresh_draw_tab(tb,in);
        draw_ls_schedule_tab_content(group[tb.now_pos]);
      handle_tab_control(tb);
      break;
    case 50:
      in=tb.now_pos;
      tb.now_pos=2;
      refresh_draw_tab(tb,in);
        draw_ls_schedule_tab_content(group[tb.now_pos]);
      handle_tab_control(tb);
      break;
    case 51:
      in=tb.now_pos;
      tb.now_pos=3;
      refresh_draw_tab(tb,in);
        draw_ls_schedule_tab_content(group[tb.now_pos]);
      handle_tab_control(tb);
      break;
    case 52:
```

```
        in=tb.now_pos;
        tb.now_pos=4;
        refresh_draw_tab(tb,in);
          draw_ls_schedule_tab_content(group[tb.now_pos]);
        handle_tab_control(tb);
        break;
      case 53:
        in=tb.now_pos;
        tb.now_pos=5;
        refresh_draw_tab(tb,in);
          draw_ls_schedule_tab_content(group[tb.now_pos]);
        handle_tab_control(tb);
        break;
      case 54:
        in=tb.now_pos;
        tb.now_pos=6;
        refresh_draw_tab(tb,in);
          draw_ls_schedule_tab_content(group[tb.now_pos]);
        handle_tab_control(tb);
        break;
      case 55:
        in=tb.now_pos;
        tb.now_pos=7;
        refresh_draw_tab(tb,in);
          draw_ls_schedule_tab_content(group[tb.now_pos]);
        handle_tab_control(tb);
        break;
      case 27:
        break;
      case 'r':
      case 'R':
        for(a=1;a<=7;a++)
          group[a]=load_group_ls_time(LS_SCHEDULE_FILE_NAME,a);
        draw_ls_schedule_tab_content(group[tb.now_pos]);
        handle_tab_control(tb);
        break;
      default:
        handle_tab_control(tb);
        break;
   }
}
int show_now_loadshedding()
{
Tab grp_tab;
int a;
for(a=1;a<=7;a++)
   group[a]=load_group_ls_time(LS_SCHEDULE_FILE_NAME,a);
ShowConsoleCursor(0);
grp_tab.lx=0;
grp_tab.ly=1;
grp_tab.height=22;
grp_tab.width=80;
grp_tab.items=7;
grp_tab.now_pos=1;
grp_tab.item_char[1]="Group-01";
grp_tab.item_char[2]="Group-02";
grp_tab.item_char[3]="Group-03";
```

```
grp_tab.item_char[4]="Group-04";
grp_tab.item_char[5]="Group-05";
grp_tab.item_char[6]="Group-06";
grp_tab.item_char[7]="Group-07";
TextColor(112);
DrawBox0(0,0,80,1,TAB_BG_TEXT);   gotoxy(1,0); putchar(240);
gotoxy(3,0);    printf("Loadshedding Schedule",254);
draw_tab(grp_tab);
TextColor(151);   DrawBox0(0,23,80,1,TAB_BG_TEXT);
TextColor(112);   DrawBox0(0,24,80,1,TAB_BG_TEXT);
gotoxy(0,24);
cprintf("\x1B",112); cprintf("ESC   R",115); cprintf("efresh     ",112);
gotoxy(0,0);
handle_tab_control(grp_tab);
getch();
}

#endif // SHOW_NOW_LS_H
```

As the source-code shows, the main functions being defined in this header are:

a. refresh_group_state()
b. draw_ls_schedule_tab_content()
c. draw_tab()
d. refresh_draw_tab()
e. handle_tab_control()

Other header files:

Other header files are also used in this program, but they are not specific for this very program, i.e they can be used for other programs also. So including all their source-code is beyond the scope of this documentation file. So, here is a list of the headers used in the program (the functions they define are described in the lower section):

1. **graphics_lib.h**
2. **DateConverter.h**
3. **xnMenu.h**

1. **graphics_lib.h:**

   The main purpose of this header is to create the graphical interface using boxes, colors and custom texts. Since we've used the Codeblocks IDE, we couldn't directly use the graphics.h header of Turbo C compiler. That's the reason we needed to create our own graphics library. Here are the functions that are defined under it:

   a. gotoxy() - sets the cursor in the value set according to the co-ordinates given
   b. Drawbox2() - draws box having double border
   c. Drawbox1() - draws box having single border
   d. Drawbox0() - draws box having no border
   e. ClearBox() - clears the drawn box
   f. ClearBoxOnly() - clears the borders of the box only
   g. DrawHorizontalLine1() - draws a horizontal line (single border)
   h. DrawHorizontalLine2() - draws a horizontal line (double border)
   i. DrawVerticalLine1() - draws a vertical line (single border)
   j. DrawVerticalLine2() - draws a vertical line (double border)
   k. cprintf() - prints texts with color
   l. TextColor() - sets a text color to the terminal (background)
   m. PrintList() - prints list to the screen
   n. ShowConsoleCursor() - hides the cursor of console
   o. Loading Animation() - loads the "Loading.." animation
   p. print_main_window() - prints the main window

2. **DateConverter.h**

   As we've targeted this program for Nepal, we used the Bikram Sambat Calendar system that is used in Nepal. For it, we needed to convert the AD calendar into the BS system. The functions used for this purpose are described briefly below:

   a. days_in_month_BS()
   b. days_in_year_BS()
   c. days_remaining_in_BS_year()
   d. days_till_in_BS_date()
   e. is_leap_year()
   f. days_in_month_AD()
   g. days_in_AD_year()
   h. days_remaining_in_AD_year()
   i. days_till_in_AD_date()
   j. difference_of_BS_dates()
   k. difference_of_AD_dates()
   l. add_days_in_AD_date()

m. add_days_in_BS_date()
n. subtract_days_in_AD_date()
o. subtract_days_in_BS_date()
p. convert_AD_date_to_BS()
q. convert_BS_date_to_AD()
r. get_now_bs_date()
s. print_nepali_date()
t. print_now_nepali_date()
u. print_now_nepali_time()
v. which_day_BS()
w. which_day_AD()
x. print_ad_date()
y. is_valid_date()

All the functions used in this header are self-descriptive.

3. xnMenu.h

The need of this header file was realized to create menu driven functions in the program. It mainly emphasizes on three functions described below:

a. print_xnMenu() - prints the menu options
b. refresh_draw_xnMenu() - refreshes the menu after performing other operations
c. handle_xnMenu() - handles the keyboard activities

Future of the Program:

We have created this simple program with basics of structures, file handling, pointers, arrays etc. At present, it just prints the loadshedding schedule as supplied by the user. But later we shall aim to make use of microprocessor programming to communicate with the power control system itself, so that we may control the entire state of power using the program. For it, we may need knowledge of C++ and hardware programming using it.

Conclusion:

Thus, with this documentation, we have revisited the program in detail. We can find that the program has been made simple graphic-wise as well as execution-wise. Using the functions available in C, we have thus made sure that any part of the programming is non-ambigious. But at the same time, we also make sure that anyone can come up with a better way of completing this task.

Any sort of suggestions and comments regarding the overall program is heartly welcome.

References:

1. “Let Us C “ - Yashwant Kanetkar
2. “Learning C by Examples " - Krishna Kandel
3. “Schaum’s outlines: Programming with C” - Byron Gottfried
4. www.lazyfoo.com
5. www.cs50.com