```c
/*############################################################################*/
/*HW06_Furkan_Erdol_131044065_part1.c                                         */
/*___________________________                                                 */
/*Written by Furkan Erdol on April 5, 2015                                    */
/*Description                                                                 */
/*____________                                                                */
/*                                                                            */
/***************This program for job assignment problem***********************/
/*<<<The aim of this program is to assign jobs to the employees as fair as    */
/*possible and show the most hardworking employee of the day and the week>>>> */
/*                                                                            */
/*-Gives inputs from enegergies file                                          */
/*-Creats schedule work schedule for employees                                */
/*-Finds the day's employee and week's employee                               */
/*-Writes report to report file                                               */
/*                                                                            */
/*Inputs:                                                                     */
/*  -Energies                                                                 */
/*Outputs:                                                                    */
/*  -Work schedule                                                            */
/*  -Day's employee                                                           */
/*  -Week's employee                                                          */
/*............................................................................*/
/*                              Includes                                      */
/*............................................................................*/
#include <stdio.h>
#define NUM_EMPLOYEES 4
#define NUM_DAYS 7

typedef enum
 {Ali, Ayse, Fatma, Mehmet}
 employee;

typedef enum
 {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}
 day;


/*Function protoytpes*/
void read_matrix(const char* file_name, int m[NUM_EMPLOYEES][NUM_DAYS]);
void create_work_plan(int job_schedule[NUM_EMPLOYEES][NUM_DAYS], int m[NUM_EMPLOYEES][NUM_DAYS]);
employee find_the_employer_of_the_day(int work_schedule[NUM_EMPLOYEES][NUM_DAYS], day day_name);
employee find_the_employer_of_the_week(int work_schedule[NUM_EMPLOYEES][NUM_DAYS]);
void report(const char* file_name, int job_scheduling[NUM_EMPLOYEES][NUM_DAYS]);
void sort_array(int array[], int size);
int max_array(const int array[], int n);
int second_max_array(const int array[], int n);
int second_min_array(const int array[], int n);
int min_array(const int array[], int n);
int index_of_max(const int array[], int n);
int index_of_second_max(const int array[], int n);
int index_of_second_min(const int array[], int n);
int index_of_min(const int array[], int n);


int
main(void)
{
    int i,j; /*For loops*/
    int job_energies[NUM_EMPLOYEES][NUM_DAYS]; /*Required energy for jobs*/
    int schedule[NUM_EMPLOYEES][NUM_DAYS]; /*Work schedule*/
    char file_energies[20]="Energies.txt";
    char file_report[20]="Report.txt";
    employee employee; /*Employee*/
    day day_name; /*Day name*/

    /*Reads job energies from file and writes it to job_energies array*/
    read_matrix(file_energies, job_energies);

    /*Reads job energies from job_energies array and creates schedule array*/
    create_work_plan(schedule, job_energies);

```

```
 73      /*Assign jobs to the employees as fair as possible*/
 74      find_the_employer_of_the_day(schedule, day_name);
 75
 76      /*Make the necessary calculations*/
 77      /*Finds day's and week's employee*/
 78      /*Finally writes report to file*/
 79      report(file_report, schedule);
 80
 81      return 0;
 82  }
 83
 84  /*Reads job energies from file and writes it to job_energies array*/
 85  void read_matrix(const char* file_name, int m[NUM_EMPLOYEES][NUM_DAYS])
 86  {
 87
 88      FILE *fp; /*File pointer*/
 89      int i,j; /*For loops*/
 90
 91      /*Open energies.txt if file couldn't open prints the screen warning message*/
 92      fp=fopen(file_name, "r");
 93      if(fp==NULL)
 94          printf("energies.txt couldn't open");
 95
 96      /*Reads energi values and creates matrix*/
 97      for(i=0;i<NUM_DAYS;i++)
 98          for(j=0;j<NUM_EMPLOYEES;j++)
 99              fscanf(fp, "%d", &m[j][i]);
100
101      fclose(fp);
102
103  }
104
105  /*Fills schedule matrix according to m array*/
106  void create_work_plan(int job_schedule[NUM_EMPLOYEES][NUM_DAYS], int m[NUM_EMPLOYEES][NUM_DAYS])
107  {
108      int i, j; /*For loops*/
109
110          for(i=0;i<NUM_EMPLOYEES;i++)
111              for(j=0;j<NUM_DAYS;j++)
112                  job_schedule[i][j]=m[i][j];
113  }
114
115  /*Make the necessary calculations*
116   *Finds day's employee and return*
117   *Finally writes report to file  */
118  employee find_the_employer_of_the_day(int work_schedule[NUM_EMPLOYEES][NUM_DAYS], day day_name)
119  {
120      int i,n,j=0; /*For loops*/
121      int sum_array[NUM_EMPLOYEES]={0}; /*Sum array for calculation*/
122      int transporter[NUM_EMPLOYEES]; /*For copying to energies*/
123      int control[NUM_EMPLOYEES]; /*To compare the arrays*/
124      employee employee; /*Employee*/
125
126      /*Fills work schedule*/
127      for(n=0;n<NUM_DAYS;n++)
128      {
129
130          for(i=0;i<NUM_EMPLOYEES;i++)
131             control[i]=work_schedule[i][n];
132
133          transporter[index_of_max(sum_array, NUM_EMPLOYEES)]=min_array(control, NUM_EMPLOYEES);
134          transporter[index_of_min(sum_array, NUM_EMPLOYEES)]=max_array(control, NUM_EMPLOYEES);
135          transporter[index_of_second_min(sum_array, NUM_EMPLOYEES)]=second_max_array(control, NUM_EMPLOYEES);
136          transporter[index_of_second_max(sum_array, NUM_EMPLOYEES)]=second_min_array(control, NUM_EMPLOYEES);
137
138          for(i=0;i<NUM_EMPLOYEES;i++)
139              sum_array[i]+=transporter[i];
140
141          for(i=0;i<NUM_EMPLOYEES;i++)
142              work_schedule[i][j]=transporter[i];
143
144          if(day_name==n)
```

```
145             employee=index_of_max(control, NUM_EMPLOYEES);
146
147     j++;
148
149     }
150
151     return employee;
152
153 }
154
155 /*Finds employer of the week according to wor schedule*/
156 employee find_the_employer_of_the_week(int work_schedule[NUM_EMPLOYEES][NUM_DAYS])
157 {
158
159     int i,j; /*For loops*/
160     int sum_array[NUM_EMPLOYEES]={0}; /*Sum array*/
161     employee employee; /*Employee*/
162
163
164     for(i=0;i<NUM_EMPLOYEES;i++)
165         for(j=0;j<NUM_DAYS;j++)
166         sum_array[i]+=work_schedule[i][j];
167
168     employee=index_of_max(sum_array, NUM_EMPLOYEES);
169
170     return employee;
171 }
172
173 /*Prints to file work schedule report*/
174 void report(const char* file_name, int job_scheduling[NUM_EMPLOYEES][NUM_DAYS])
175 {
176     FILE *fp;
177     int i,j;
178     char days[NUM_DAYS][13]={"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};
179     char employees[NUM_EMPLOYEES][13]={"Ali", "Ayse", "Fatma", "Mehmet"};
180     employee employee;
181
182     fp=fopen(file_name, "w");
183
184     fprintf(fp, "Report:\n\n%7c", ' ');
185     for(i=0;i<NUM_DAYS;i++)
186         fprintf(fp, "%-10s",  days[i]);
187
188     for(i=0;i<NUM_EMPLOYEES;i++)
189     {
190         fprintf(fp, "\n\n%-9s",employees[i]);
191         for(j=0;j<NUM_DAYS;j++)
192             fprintf(fp, "%-10d",job_scheduling[i][j]);
193     }
194
195
196     employee=find_the_employer_of_the_day(job_scheduling, Monday);
197     fprintf(fp, "\n\nThe best employer of Monday : ");
198     switch(employee)
199     {
200
201     case 0 : fprintf(fp, "Ali");
202              break;
203     case 1 : fprintf(fp, "Ayse");
204              break;
205     case 2 : fprintf(fp, "Fatma");
206              break;
207     case 3 : fprintf(fp, "Mehmet");
208              break;
209
210     }
211
212     employee=find_the_employer_of_the_day(job_scheduling, Tuesday);
213     fprintf(fp, "\n\nThe best employer of Tuesday : ");
214     switch(employee)
215     {
216
```

```
217       case 0 : fprintf(fp, "Ali");
218                break;
219       case 1 : fprintf(fp, "Ayse");
220                break;
221       case 2 : fprintf(fp, "Fatma");
222                break;
223       case 3 : fprintf(fp, "Mehmet");
224                break;
225
226       }
227
228       employee=find_the_employer_of_the_day(job_scheduling, Wednesday);
229       fprintf(fp, "\n\nThe best employer of Wednesday : ");
230       switch(employee)
231       {
232
233       case 0 : fprintf(fp, "Ali");
234                break;
235       case 1 : fprintf(fp, "Ayse");
236                break;
237       case 2 : fprintf(fp, "Fatma");
238                break;
239       case 3 : fprintf(fp, "Mehmet");
240                break;
241
242       }
243
244       employee=find_the_employer_of_the_day(job_scheduling, Thursday);
245       fprintf(fp, "\n\nThe best employer of Thursday : ");
246       switch(employee)
247       {
248
249       case 0 : fprintf(fp, "Ali");
250                break;
251       case 1 : fprintf(fp, "Ayse");
252                break;
253       case 2 : fprintf(fp, "Fatma");
254                break;
255       case 3 : fprintf(fp, "Mehmet");
256                break;
257
258       }
259
260       employee=find_the_employer_of_the_day(job_scheduling, Friday);
261       fprintf(fp, "\n\nThe best employer of Friday : ");
262       switch(employee)
263       {
264
265       case 0 : fprintf(fp, "Ali");
266                break;
267       case 1 : fprintf(fp, "Ayse");
268                break;
269       case 2 : fprintf(fp, "Fatma");
270                break;
271       case 3 : fprintf(fp, "Mehmet");
272                break;
273
274       }
275
276       employee=find_the_employer_of_the_day(job_scheduling, Saturday);
277       fprintf(fp, "\n\nThe best employer of Saturday : ");
278       switch(employee)
279       {
280
281       case 0 : fprintf(fp, "Ali");
282                break;
283       case 1 : fprintf(fp, "Ayse");
284                break;
285       case 2 : fprintf(fp, "Fatma");
286                break;
287       case 3 : fprintf(fp, "Mehmet");
288                break;
```

```
289
290       }
291
292       employee=find_the_employer_of_the_day(job_scheduling, Sunday);
293       fprintf(fp, "\n\nThe best employer of Sunday : ");
294       switch(employee)
295       {
296
297       case 0 : fprintf(fp, "Ali");
298                break;
299       case 1 : fprintf(fp, "Ayse");
300                break;
301       case 2 : fprintf(fp, "Fatma");
302                break;
303       case 3 : fprintf(fp, "Mehmet");
304                break;
305
306       }
307
308       employee=find_the_employer_of_the_week(job_scheduling);
309       fprintf(fp, "\n\nThe best employer of the week is ");
310       switch(employee)
311       {
312
313       case 0 : fprintf(fp, "Ali ... Congratulations Ali !!");
314                break;
315       case 1 : fprintf(fp, "Ayse ... Congratulations Ayse !!");
316                break;
317       case 2 : fprintf(fp, "Fatma ... Congratulations Fatma !!");
318                break;
319       case 3 : fprintf(fp, "Mehmet ... Congratulations Mehmet !!");
320                break;
321
322       }
323
324   }
325
326   /*Gives an array and sort it*/
327   void sort_array(int array[], int size){
328
329       int i, k; /*For loops*/
330       int temp; /*Temporary variable*/
331
332       for(i=0;i<size;i++)
333       {
334           for(k=i+1;k<size;k++)
335           {
336               if(array[i]<array[k])
337               {
338                   temp=array[i];
339                   array[i]=array[k];
340                   array[k]=temp;
341               }
342           }
343        }
344   }
345
346   /*Finds maximum number in array and return it*/
347   int max_array(const int array[], int n)
348   {
349
350       int i,
351           max=array[0];/*Maximum number in array*/
352
353       for(i=0;i<n;i++)
354           if(array[i]>max)
355               max=array[i];
356
357       return max;
358   }
359
360   /*Finds second maximum number in arrayand return it*/
```

```c
int second_max_array(const int array[], int n)
{

    int i,
        max, /*Maximum number in array*/
        second_max=0; /*Maximum second number in array*/
    int counter=0;

    max=max_array(array, n); /*Calls max array function for give maximum number*/

    for(i=0;i<n;i++)
    {
        if(array[i]>second_max&&array[i]<max)
            second_max=array[i];
        else if(array[i]==max)
            counter++;
        }

    if(counter>1)
        second_max=max;

    return second_max;

}

/*Finds second minimum number in array and return it*/
int second_min_array(const int array[], int n)
{

    int i,
        min, /*Minimum number in array*/
        second_min=max_array(array, n); /*Minimun second number in array*/
    int counter=0;

    min=min_array(array, n); /*Calls min array function for give minimum number*/

    for(i=0;i<n;i++)
    {

        if(array[i]<second_min&&array[i]>min)
            second_min=array[i];
        else if(array[i]==min)
            counter++;

    }

    if(counter>1)
        second_min=min;

    return second_min;

}

/*Finds minimum number in array and return it*/
int min_array(const int array[], int n)
{

    int i,
        min=array[0]; /*Minimum number in array*/


    for(i=0;i<n;i++)
        if(array[i]<=min)
            min=array[i];

    return min;
}

/*Finds maximum number's index in array and return it*/
int index_of_max(const int array[], int n)
{

```

```
433      int i,
434          max=array[0], /*Maximum number in array*/
435          index=0; /*Maximum number's index*/
436
437      for(i=0;i<n;i++)
438          if(array[i]>=max)
439          {
440              max=array[i];
441              index=i;
442          }
443
444      return index;
445  }
446
447  /*Finds second maximum number's index in array and return it*/
448  int index_of_second_max(const int array[], int n)
449  {
450
451      int i,
452          max, /*Maximum number in array*/
453          second_max=0, /*Second maximum number in array*/
454          index=0, /*Second maximum number's index*/
455          control_position=0;
456      int counter=0;
457
458      max=max_array(array, n); /*Calls max array function for give maximum number*/
459
460      for(i=0;i<n;i++)
461      {
462          if(array[i]>=second_max&&array[i]<max)
463          {
464              second_max=array[i];
465              index=i;
466          }
467          else if(array[i]==max)
468          {
469              counter++;
470              if(counter==2||counter==3||counter==4)
471              {
472                  second_max=max;
473                  index=control_position;
474              }
475              control_position=i;
476          }
477      }
478
479      return index;
480
481  }
482
483  /*Finds second minimum number's index in array and return it*/
484  int index_of_second_min(const int array[], int n)
485  {
486
487      int i,
488          min, /*Minimum number in array*/
489          second_min=max_array(array, n), /*Second minimum number in array*/
490          index=0; /*Second minimum number's index*/
491      int counter=0;
492
493      min=min_array(array, n); /*Calls min array function for give minimum number*/
494
495      for(i=0;i<n;i++)
496      {
497
498          if(array[i]<second_min&&array[i]>min)
499          {
500              second_min=array[i];
501              index=i;
502          }
503          else if(array[i]==min)
504          {
```

```
505             counter++;
506             if(counter==2)
507             {
508                 second_min=min;
509                 index=i;
510             }
511         }
512     }
513 
514     return index;
515 }
516 
517 /*Finds minimum number's index in array and return it*/
518 int index_of_min(const int array[], int n)
519 {
520 
521     int i,
522         min=array[0]; /*Minimum number in array*/
523     int index=0; /*Minimum number's index*/
524 
525     for(i=0;i<n;i++)
526         if(array[i]<min)
527         {
528             min=array[i];
529             index=i;
530         }
531 
532     return index;
533 }
534 
535 /*##############################################################################*/
536 /*                End of HW06_Furkan_Erdol_131044065_part1.c                    */
537 /*##############################################################################*/
```