```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_LEN 100

struct Date_t {
  int day;
  char mon[4];
  int year;
};
typedef struct Date_t Date;


struct Event_t {
  char *desc;
  struct Event_t *next;
};
typedef struct Event_t Event;


struct HistoricalDate_t {
  Date dt;
  Event *elist;
  struct HistoricalDate_t *next;
};
typedef struct HistoricalDate_t HistoricalDate;

FILE *parse_parameters(int argc,
                       char *argv[]);
HistoricalDate *read_historical_events(
                                    FILE *file);
int read_historical_event(FILE *file,
                       Date *pdt, char *desc);
int get_date(FILE *file, Date *pdt);
HistoricalDate *find_historical_date(
               HistoricalDate *head, Date dt);
int dates_equal(Date dt1, Date dt2);
int add_event(HistoricalDate *hdate,
              char *desc);
void free_historical_dates(
                        HistoricalDate *head);
void free_event_list(Event *head);
void print_events_for_date(
               HistoricalDate *head, Date dt);
void print_events(Event *head);
```

int main(int argc, char *argv[])

```c
{
  FILE *file;
  HistoricalDate *hist;
  Date dt;

  file = parse_parameters(argc, argv);
  hist = read_historical_events(file);
  if(hist == NULL) return 0;

  printf("Enter dates:\n");
  while(get_date(stdin, &dt)) {
    print_events_for_date(hist, dt);
  }

  free_historical_dates(hist);
  return 0;
}
```

FILE *parse_parameters(int argc, char *argv[])

```c
{
  FILE *file = NULL;
  if(argc!=2) {
    fprintf(stderr, "usage err\n");
    exit(1);
  }

  file = fopen(argv[1], "r");
  if(file==NULL) {
    fprintf(stderr, "Cannot read\n");
    exit(2);
  }
  return file;
}
```

HistoricalDate *read_historical_events(FILE *file)

```c
{
  Date dt;
  char desc[MAX_LEN];
  HistoricalDate *head = NULL;
  HistoricalDate *hdate;

  while(read_historical_event(file,&dt,desc)){
    hdate = find_historical_date(head, dt);
    if(!hdate) {
      hdate = malloc(sizeof(HistoricalDate));
      hdate->dt = dt;
      hdate->elist = NULL;
      hdate->next = head;
      head = hdate;
    }
    if(add_event(hdate, desc)!=0) {
      free_historical_dates(head);
      return NULL;
    }
  }
  return head;
}
```

int read_historical_event(FILE *file, Date *pdt, char *desc)

```c
{
  if(pdt==NULL || desc==NULL) return 0;

  if(!get_date(file, pdt)) {
    return 0;
  }
  fscanf(file, " "); /* skip spaces */
  if(!fgets(desc, MAX_LEN, file)) {
    return 0;
  }
  if(desc[strlen(desc)-1]=='\n') {
    desc[strlen(desc)-1]='\0';
  }
  return 1;
}
```

int get_date(FILE *file, Date *pdt)

```c
{
  if(pdt == NULL) return 0;
  return fscanf(file, "%d %s %d",
                &(pdt->day),
                pdt->mon,
                &(pdt->year))==3;
}
```

HistoricalDate *find_historical_date(
HistoricalDate *head, Date dt)

```
{
  while(head) {
    if(dates_equal(head->dt, dt)) return head;
    head = head->next;
  }
  return NULL;
}
```

int dates_equal(Date dt1, Date dt2)

```
{
  return (dt1.day==dt2.day) &&
         (strcmp(dt1.mon, dt2.mon)==0) &&
         (dt1.year==dt2.year);
}
```

int add_event(HistoricalDate *hdate, char *desc)

```
{
  Event *ev;

  if(hdate==NULL || desc==NULL) return 1;

  ev = malloc(sizeof(Event));
  if(ev == NULL) return 2;

  ev->desc =
        malloc((strlen(desc)+1)*sizeof(char));
  if(ev->desc == NULL) {
    free(ev);
    return 3;
  }

  strcpy(ev->desc, desc);
  ev->next = hdate->elist;
  hdate->elist = ev;

  return 0;
}
```

void free_historical_dates(HistoricalDate *head)

```
{
  HistoricalDate *tmp;
  while(head) {
    tmp = head;
    head = head->next;
    free_event_list(tmp->elist);
    free(tmp);
  }
}
```

void free_event_list(Event *head)

```
{
  Event *tmp;

  if(head == NULL) return;

  while(head) {
    tmp = head;
    head = head->next;
    free(tmp->desc);
    free(tmp);
  }
}
```

void print_events_for_date(HistoricalDate *head,
Date dt)

```
{
  HistoricalDate *hdate;

  hdate = find_historical_date(head, dt);
  if(hdate == NULL || hdate->elist == NULL) {
    printf("Nothing special\n");
    return;
  }

  print_events(hdate->elist);
}
```

void print_events(Event *head)

```
{
  while(head) {
    printf("%s\n", head->desc);
    head = head->next;
  }
}
```