McMaster University
Department of Computing and Software
Dr. W. Kahl

SFWR ENG 2S03 — Principles of Programming

18 October 2006

Exercise 6.1 — Calendar (22% of Final 2003)

For a calendar application, a year will be represented by **a single contiguous array** of days, called a **"year array"**.

For making access easier, a **"month start array"** will be calculated, containing for each month index *i* the index that the first day of month *i* has in year arrays.

Example: In a normal (i.e., non-leap) year, the first four elements (at indices 0, 1, 2, 3) of the month start array will be 0, 31, 59, 90.

Note: The items (a) and (b) are completely independent of each other.

(a) ≈10% Implement the C function

int * *startDays*(int *monthsNum* , const int *monthLen*[] , int * *yearLen*)

that

- returns a pointer to the beginning of a newly allocated month start array which should have *monthsNum* elements,
- initialises this new month start array according to the month lengths found in the *monthsNum*-element array *monthLen*, and
- writes the number of days the whole year has in this calendar into the reference parameter *yearLen*.

(b) ≈12% Implement the iterative C function

void *printDate*(int *monthsNum* , int *monthStart*[] , int *index*)

that, given a number of months and a month start array, uses **binary search** to find the month containing the day with index *index* in a year array; it should then print (to standard output) a message containing the day in that month and the number of the month as user-level day and month numbers.

Example: For index 0 it should print "`Day 1 month 1`", and for index 33 (using the standard calendar) it should print "`Day 3 month 2`".

Let the following enumeration type definition be given:

typedef enum {*SUN*, *MON*, *TUE*, *WED*, *THU*, *FRI*, *SAT* } *Weekday*;

(c) new Write a C function *weekday* that, given a month start array *monthStart*, the weekday *wd1* of the first day of the year (for 2003 this would be *WED*), and two int values *month* and a *day*, returns the weekday of the day indicated by *month* and a *day*, which are supplied as user-level numbers: For the 21st October, these arguments would be *month*=10 and *day*=21.

Exercise 6.2 — Calendar (modified 23% of Final 2003)

For the calendar application of Exercise 6.1:

(a) Write and document **appropriate** type definitions for the calendar data — of type *Day* — to be stored in year arrays.

For each day, there should be the times of sunrise and sunset, and up to 10 appointments.

An appointment — of type *Appointment* — has begin and end times, a title string, and a comment string.

(b) **Design and implement** a C function *find* that accepts the following parameters:
- the number of months and a month start array,
- the number of days in the year and a year array containing *Day* elements,
- a **function** *check* that takes an *Appointment* — see (c) — as argument and returns either *NULL* to signal that the argument *Appointment* is irrelevant, or a pointer to a string containing a message to be printed.

The function *find* should apply *check* to all appointments in the year array, and for each appointment for which a message is returned, it should print the message and use *printDate* from (b) above to print the date at which the appointment was found.

(c) new Implement argument functions for *find* from (b), e.g.:
- *checkWhite* finds appointments where the comment string contains only white-space characters, and returns a message transscribing the comment into a C string literal.

  So if the comment consisted of an empty line, and a line containing a space and a tab character, the returned message, when printed to the screen, would contain the nine-character string "\n \t\n".

  (For manually generating this, you would write: "\"\\n \\t\\n\"".)
- *checkBirthday* finds birthdays: If the comment of an appointment does not contain (case insensitive) the sub-string "birthday", it returns *NULL*. If a birthday comment starts with "Birthday: ", then *checkBirthday* only returns the suffix after that prefix, otherwise the whole comment.

(d) new Write a *main* program to test everything!

Exercise 6.3 — Typing (22% of Midterm 2, 2005)

Give variable declarations (and only variable **declarations**) to preceed the following statements so that the resulting code is valid ANSI C. In each case, you must provide **the most appropriate type**.

(a)
```
d = 0.5;
```

(b)
```
*p = q + 0.5;
```

(c)
```
p = q + *q;
```

(d)
```
array[3] = 3.14;
```

(e)
```
*answer = 42;
```

(f)
```
array = malloc( 10 * sizeof (double) );
array[6] = 2.73e5;
```

(g)
```
matrix = malloc( 5 * sizeof (double *) );
matrix[2] = malloc( 8 * sizeof (double) );
matrix[2][4] = 0.0;
```