Structures

1. Write a function elapsed\_time that takes as its arguments two time structures and returns a time structure that represents the elapsed time (in hours, minutes, and seconds) between the two times. So the call elapsed\_time (time1, time2) where time1 represents 3:45:15 and time2represents 9:44:03, should return a time structure that represents 5 hours, 58 minutes, and 48 seconds. Be careful with times that cross midnight.
2. Write a function called clockKeeper that takes as its argument a dateAndTime structure as defined in this chapter.The function should call thetimeUpdate function, and if the time reaches midnight, the function should call the dateUpdate function to switch over to the next day. Have the function return the updated dateAndTime structure.
3. Write a program that manages a simple "database" of students and their marks.

The program reads from stdin information about n students and stores them in an array of strucures.

The program prints then the list of students in alphabetic order of their names.

The program should be composed from 3 functions: read\_students(), sort\_students() and print\_students().

Use following global data structures:

#define MAXSTUD 20

struct student {

char \*name;

float mark;

};

struct student a[MAXSTUD];

int nstud;

1. Modifyprogram 2 to use the following data structure:

struct student {

char \*name;

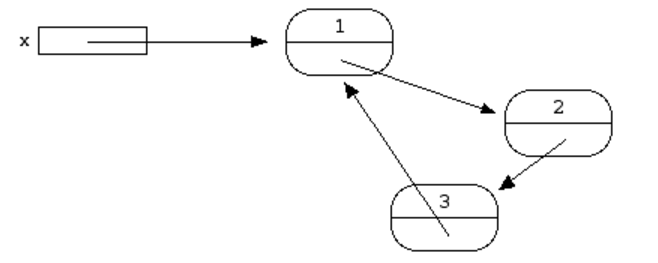
float mark;

};

struct student \*a; // dynamically allocated array of structures

int nstud;

1. (Array of structures.)Write a program that:
2. reads a set of calendar dates from standard input and stores them into an array
3. sorts them in chronological order and prints the sorted list
4. reads a date currentDate from standard input and splits the initial array of dates into 2 arrays, one array that contains only the dates that are before currentDate, and one array that contains only the dates that are after currentDate
5. Suppose you have a pointee type called "Node" which contains two things: an int, and a pointer to another Node (the declaration for such a Node type is given below). With such a pointee type, you could arrange three Node pointees in a structure where they were pointing to each other like this...



The pointer named x points to the first Node pointee. The first Node contains a pointer to the second, the second contains a pointer to the third, and the third contains a pointer back to the first. This structure can be build using only the rules of pointee allocation, dereferencing. Using the declaration below, each Node contains an integer named value and a pointer to another Node named next.

struct Node {

int value;

struct Node\* next;

};

Write the code to build the structure in the above drawing. For convenience, you may use temporary pointers in addition to x. The only new syntax required is that in C, the operator -> dereferences a pointer to access a field in the pointee – so ->value accesses the field named value in x's pointee.

1. The Sky Airlines fleet consists of P=5 planes with a seating capacity of N=10 seats each. Each of them makes one flight daily. Write a seating reservation program with the following features:
   1. The program uses arrays of structures. Each structure should hold a seat identification number, a marker that indicates whether the seat is assigned, the social security number (CNP) of the person assigned there.
   2. The program displays the following menu:

1)choose plane number

2)quit

If the user enters 1, he will be asked to enter a valid plane number (1-p) and a second menu is displayed:

To choose a function, enter its letter label:

a) Show number of empty seats

b) Show list of empty seats

c) Assign a customer to a seat assignment

d) Delete a seat assignment

e) return to main menu

c. The program successfully executes the promises of its menu. Choices c) and d) require additional input.

d. After executing a particular function, the program shows the menu again