CSCI 2100B Data Structures

Assignment 1
Due Date: 15 February 2022

Written Exercises

There is no written exercise.

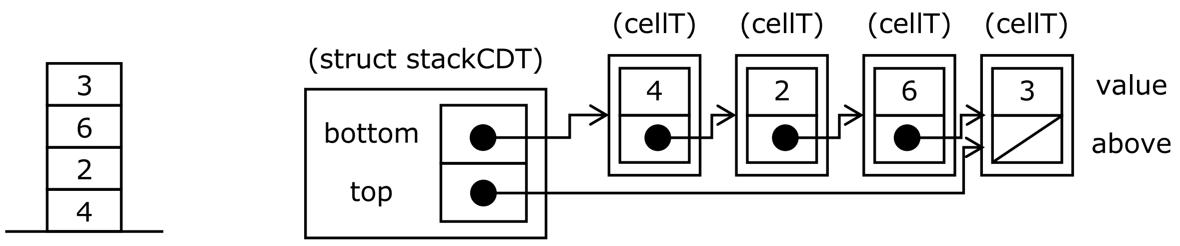
Programming Exercises

1. In the lecture we have completed a concrete implementation using linked lists for stack ADT. In this exercise we are going to try two new implementations of the stack ADT.

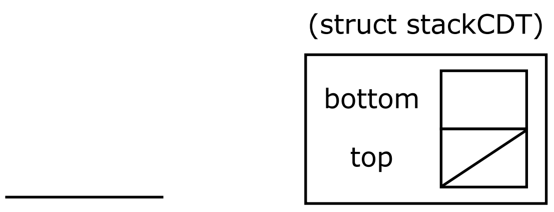
   The first new implementation will be called version 5.0. The main idea is to implement a stack using a linked list. Specifically, assuming that all stack elements are integers, we write in "stack.c" the following concrete implementation

```
#include "stack.h"
/* version 5.0 */

typedef struct cellT {stackElementT value; struct cellT *above;} cellT;

struct stackCDT {
  cellT *bottom;
  cellT *top;
};
```

   The following diagrams show the implementation of a nonempty stack:



   The following diagrams show the implementation of an empty stack:

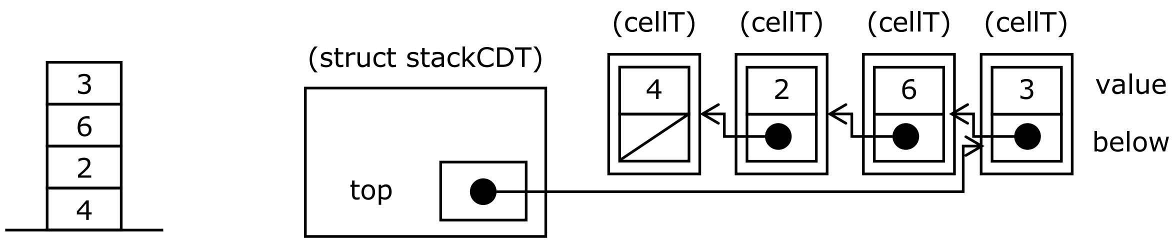


   Complete the implementation version 5.0.

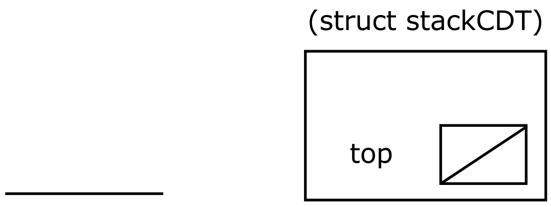
2. If we carefully examine the implementation version 5.0 of stackADT, it can be concluded that the design is bad. First, the field bottom in struct stackCDT is not necessary. In addition, the direction of the pointers in the linked list should be reversed. One can write the improved implementation, which will be called version 5.1.

```
#include "stack.h"
/* version 5.1 */

typedef struct cellT {stackElementT value; struct cellT *below;} cellT;

struct stackCDT {
  cellT *top;
};
```

The following diagrams show the implementation of a nonempty stack:



The following diagrams show the implementation of an empty stack:



Complete the implementation version 5.1.

3. Write a program to simulate the following scenario. You should use the "queue.h" header file defined in the lecture notes. Your program should work well with the implementation shown in the lecture notes, as well as any correct implementation. (Note also that you need to appropriately modify the queueElementT.)

   We consider a small bus terminus, from which three bus routes, namely, route A, route B and route C, start. At each of the bus stops of routes A, B and C, there is an LCD display board, on which the time the next bus leaves as well as the bus fare will be displayed. If the last bus has departed, then the display is turned off. Note that the bus fare of a bus route can be different at different time of a day. Passengers waiting for the buses form a queue at each of the bus stops.

   We define a display board as a structure:

```
typedef struct displayBoard {
   int departureTime = -1;
   int busFare = 0;
} displayBoard;

displayBoard boardA, boardB, boardC;
```

   We define a bus stop as a structure:

```
typedef struct busStop {
   stackADT departureTime;
   stackADT busFare;
   queueADT passengerQ;
} busStop;

busStop stopA, stopB, stopC;
```

The member `departureTime` of `busStop` is a stack of integers that stores the departure times of a bus route, expressed in terms of the number of minutes since 0:00 of a day. The member `busFare` is a stack of integers that stores the different bus fares corresponding to the departure times stored in `departureTime`. Therefore, the heights of both stacks are always the same.

Finally, `passengerQ` is a queue of `PassengerADT` type defined as follows.

```
typedef struct Passenger {
   int PassengerClass; /* 0: child; 1: adult; 2: senior citizen */
   int routeAOK;
   int routeBOK;
   int routeCOK;
   int timeEnqueued;
} Passenger;

typedef Passenger *PassengerADT;
```

Note that a passenger can be a child, an adult, or a senior citizen. A child's bus fare is always half of the full fare, rounded down to the nearest dollar. A senior citizen's fare is either half of the full fare rounded down to the nearest dollar, or $2, whichever is the lower. If a passenger can take a route A bus, then `routeAOK` should be set to 1, otherwise it should be zero. The same is true for `routeBOK` and `routeCOK`. Finally, `timeEnqueued` records the time the passenger joins the queue.

Now write a C program to simulate the operations of the bus terminus in a day from 0:00 to 23:59. The main program consists in a loop, in which a control variable (say `time`) takes the integer values from 0 (representing 0:00) to 1439 (representing 23:59).

INITIALISATION

Before the loop is executed, the variables `stopA`, `stopB` and `stopC` representing the three bus stops are declared and initialised as follows:

- `departureTime` is first initialised to be an empty stack. Then the departure times of the buses are read from three text files 'departureA.txt', 'departureB.txt', and 'departureC.txt'. Note that the earliest departure time should be stored at the top of the stack `departureTime`. We assume that the departure times stored in these files are always shown in ascending order of time. For example, file 'departureA.txt' contains the following lines

```
480
510
540
570
600
630
```

That means for route A, there are totally 6 departures from 8:00 to 10:30, with a headway of 30 minutes.

- `busFare` is first initialised to be an empty stack. Then the full fare of the buses are read from three text files 'fareA.txt', 'fareB.txt', and 'fareC.txt'. For example, if the file 'fareA.txt' contains the following lines

```
10
10
10
8
8
8
```

That means for route A, the full fare for the buses departing at 8:00, 8:30 and 9:00 is $10, while that for the buses departing at 9:30, 10:00 and 10:30is $8.

- `passengerQ` is initialised to be an empty queue. Note that passenger arrive at the bus terminus from time to time, and the information is stored in the file 'PassengerArrival.txt'. The following shows a portion of an example file:

```
460 1 A
460 1 A
460 2 B A
461 0 A C
479 1 C
481 1 C
483 0 C
483 0 C
483 1 C
483 2 C
483 0 A C
```

The information shows
- At 7:40, an adult passenger arrives. The passenger takes bus route A.
- At 7:40, an adult passenger arrives. The passenger takes bus route A. (Note that this passenger is a different passenger from the first one.)
- At 7:40, a senior citizen passenger arrives. The passenger takes bus route A or route B.
- At 7:41, a child passenger arrives. The passenger takes bus route A or C.
- At 7:59, an adult passenger arrives. The passenger takes bus route C.
- At 8:01, an adult passenger arrives. The passenger takes bus route C.
- At 8:03, a child passenger arrives. The passenger takes bus route C.
- At 8:03, a child passenger arrives. The passenger takes bus route C.
- At 8:03, an adult passenger arrives. The passenger takes bus route C.
- At 8:03, a senior citizen passenger arrives. The passenger takes bus route C.
- At 8:03, a child passenger arrives. The passenger takes bus route A or C.

SIMULATION

The execution of the loop body simulates what happens in one minute. The following points highlight the main activities in one minute. Remember that the current time is represented by the value of the variable `time`.

(1) Update the display boards using the information in the two stacks in a bus stop if necessary.

(2) If necessary, read information from 'PassengerArrival.txt'.

   a. When a passenger arrives, he joins the corresponding queue immediately. However, if he can take more than one route, he will join the shortest queue. For example, if a passenger can take either route A or route B, and the current length of the queue at bus stop A is shorter than that at bus stop B, then he joins the queue at bus stop A. If the lengths of the two queues are exactly the same, then the passenger joins the queue for route with a lower bus fare. Finally, if the lengths of the queues and bus fares are all the same, then a passenger prefers route A to route B, and route B to route C.

   b. Remember to initialise/ update the variables and members of structures.

(3) If a bus should depart at the current time, then all passengers in the corresponding queue get on the bus, and the queue becomes an empty queue. We assume that the buses are always big enough for all passengers in the queue to get on.

OUTPUT

At the end of the simulation, you should print the following data.

1. The total number of passengers for routes A, B and C.

2. The total number of bus fare received for routes A, B and C.

3. The average waiting time for passengers of routes A, B and C. Note that if a passenger joins the queue at t_1, and gets on the bus at t_2, then the waiting time is $t_2 - t_1$.

End of Assignment