ROB-GY 6413
Robots for Disability

Simulation of an Automatic pill dispenser using C language

Project Report submitted to Professor Kapila in partial fulfillment of the requirements of the course

Arunagiri Adhithian R - ar7404

Erin Butler - ecb9931

Nithya Muralidaran - nm3671

November 2022

NYU | TANDON SCHOOL OF ENGINEERING

TABLE OF CONTENTS

Abstract

If various medication regimens are not adhered to, it causes effects from worsening of disease to even death. These effects are more prevalent in patients with memory impairment. This barrier can be addressed by the usage of automatic pill dispensers, whose functionality is simulated in this project. This project emulates the functionality of an automatic pill dispenser that allocates pills with prescribed timings of medication and various methods of notifying the patients that they need to take their medication.

Objective

The goal of this simulation project is to simulate the working of an automatic pill dispenser that reduces assistance from caregivers and increases the independence of the patient. This simulation will be completed in C programming language with print statements indicating at which state the simulation is in.

Background

Medication adherence is critical for patient recovery and overall treatment cost reduction. Medication adherence has the potential to reduce disease-related medical costs and hospitalization rates significantly. Medication adherence is not unique to elderly or mentally impaired patients. Krueger et al. (2015) carried out a systematic review. [1] A study of patient medication adherence discovered that younger people are more likely than older people to skip doses. Similarly, Jasti et al. (2005) studied multivitamin pill adherence based on socioeconomic characteristics and discovered that adherence was low among low-income, less-educated individuals. McDonald, Garg, and Haynes (2002) identified the need for novel approaches to assist patients in adhering to medication prescriptions. [2]

Many researchers have echoed this need, believing that while there has been tremendous progress in designing interventions for medication adherence, the level of adherence to medications is still not acceptable, particularly in low-income, less-educated families. The goal of this project is to simulate a low-cost system that can automatically dispense medication. Individuals, particularly the cognitively impaired, children, bedridden patients, and the less educated, will find it useful. [3]

Pill boxes are a popular way to remind people about their medication schedules. A plaid-based pill box is the most common type. It is used to store medications on a schedule, such as daily pill boxes, weekly pill boxes, weekly four-times pill boxes, and so on. Furthermore, the pill dispenser aids the patient in remembering to take their medication on time. There are various types of pill dispensers on the market that have been manufactured by various companies. GMS MED-E-LERT automatic pill dispenser, E-Pill electronic dispenser, E-tamper-proof Pill's automatic medication dispenser, and Philips medication dispensing service are just a few examples. [4]

The only pill dispensers on the market have built-in alarms for medication time notifications. Due to hearing loss, it is difficult for elderly patients to hear the alarm, especially when the pill dispenser is placed far away from them. Other researchers have recently made numerous changes to the pill box. For example, the proposed Med Tracker electronic pill box can record the time medicine is taken from the box.[5] Then, a smart pill box is proposed, which includes reminder and confirmation functions via the use of a matrix barcode printed on the medicine bag. Finally, as previously reported, an intelligent pill box proposed a medicine bag system with a notification system utilizing the Skype application.

Code Flow

When the simulation is started, the user is walked through an initial prescription input function to input how many pills the patient will take, on which days and at which times the pills need to be dispensed, and how many pills are loaded. This information is sorted into arrays. A while loop will then be used to track the state of the system, outputting various print statements signifying what is happening. Different functions, explained in the comments of the code below are called to progress through different states.

First the current time is updated to store values in ‘day’, ‘hour’ and ‘min’ variables. It is assumed that pills are only dispensed on the hour mark. If the minute value is 0, the current time is compared to the prescription arrays. If there are no pills to be dispensed at this hour, a print statement shows this and the simulation continues.

If there are any pills to dispense at that time, a print statement shows which tracks are dispensing a pill. The system then moves into a notification state. For the first notification, a print statement shows that an LED is flashing. The user has five minutes to enter that they have taken their medication. In the demonstration, this time is shortened to five seconds. If no input is detected, the second notification is a solid LED with a buzzer sounding every 15 seconds. Again the user has to press enter when they have taken the medication. If no input is detected, the third notification keeps the LED and buzzer on and sends a notification to the caregiver. If after all three notifications the patient still has not taken their medication, it is assumed that the caregiver will ensure their medication is taken in order to allow the dispenser to continue monitoring the time for the next dispensation.

As pills are dispensed, the count of pills remaining is updated. Afterwards, the pill counts are checked to ensure there is enough medication available. If any of the pill counts drops below 10, a notification is sent to the caregiver alerting them to refill the dispenser.

The dispenser also allows the user to enter updates. If the minute is less than 55, the user can enter the following characters to make the appropriate updates to the pill dispenser and prescription.

- U: Update a current prescription
- R: Refill a current prescription
- N: Add a new prescription
- D: Delete a current prescription

A buffer of 5 minutes is used as the time begins to approach the hour mark. This buffer ensures that no checks for pill dispensing are missed.

Two versions of the code are uploaded: “PillDispenserFinal.c” and “PillDispenserDemo.c”. The only difference between these two codes is how the updateNow function is written. This function updates the current time variables (day, hour, and minute). In the Final version, the computer time is used to update the current time. This is the code that would be used on the actual device. However, showing this as a simulation would not be productive because it would move too slowly through the states. The Demo version uses a timeCounter to increment through each minute of the day. There are 1,440 minutes every day. The counter increments once each time the function is called. The minute is found as the counter modulus 60. If the result is 0, an hour has elapsed and the hour variable is incremented while the minute variable is reset. When the count reaches 1,440, the day is incremented and both the minute and hour are reset. Once 7 days have elapsed, the day is reset to 1, and the simulation repeats. Simulating time in this fashion allows for a faster simulation.

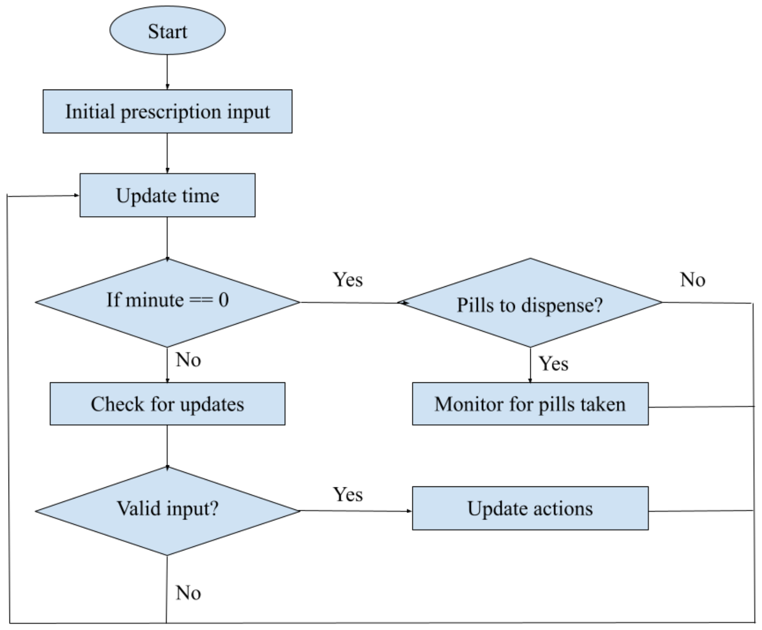

Figure 1: Flowchart for Automatic Pill Dispenser Simulation

Results

A demonstration video, ‘Automatic Pill Dispenser Demo.mp4’ has been uploaded which goes through each of the components of the simulation. Pauses are used throughout the code to allow time for the viewer to see what is happening, however, the video can be paused if more time is needed. A ‘Demo Input.txt’ Supports this video, outlining in order the various control functions demonstrated.

Impact

A successful simulation of this pill dispenser will provide evidence for creating a working, low-cost prototype that will be accessible to a wider range of caregivers and patients. By decreasing the burden of medication management on the caregiver, more time can be devoted to other activities to benefit either the patient or the caregiver. With the help of automatic pill dispensing, caregivers will have less worry associated with ensuring their loved ones are cared for appropriately, especially in situations where they are unable to be physically present. Improving feelings of independence for the patient can result in an improved emotional state. Giving the patient multiple opportunities to take their medication on their own can be a stimulating task, helping to sharpen their cognitive and motor abilities. In general, an automatic pill dispenser can reduce the risk of overdose or missed medications for a range of patients.

Code

The following are assumptions made when simulating the pill dispenser.

- A maximum of 20 pills would be prescribed for a patient. This can be easily changed by updating MAX_PILLS definition. When the prescription arrays are printed to demonstrate the update actions, only the first five pills will be displayed for brevity.
- Pills will only be dispensed at the hour mark. This is typical of prescriptions.
- After 3 notifications the caregiver will ensure pills are taken to make sure no future pill dispense checks are missed.

Below is the code used for this simulation (PillDispenserFinal.c). See the updateNow function for both ways of determining the current time.

```
/*
Erin Butler, Nithya Muralidaran, Arunagiri Ravichandran
Robots for Disability - ROBGY 6413
Simulation Project: Automatic Pill Dispenser
Fall 2022
*/

/*
v5 cleans up print statements for easier reading during the demo. More
comments are added.
```

```
v4, when pills are dispensed pillCounts, is updated and a warning
      is sent to the caregiver to refill
v3 when not dispensing, allowing the user to make edits to prescriptions.
v2 allows for pill profile entry and monitors dispensing and
      pill status when min = 0.
v1 includes all user input functions for entering a pill profile.
*/

//-----------------------------------------------------------------------
// Include needed header files.
#include <stdio.h>
#include <time.h>
#include <windows.h> // Getting current time.
#include <unistd.h> // Using timeout functions.
#include <sys/time.h>
#include <sys/types.h>
#include <stdlib.h> // Using system("clear"); to clear terminal.
/*
Sources:
Obtaining current computer time.
https://www.geeksforgeeks.org/time-h-header-file-in-c-with-examples/
Waiting a certain amount of time for user input (timeout function).
https://stackoverflow.com/questions/7226603/timeout-function
*/
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
// Initialize variables and functions for storing pill profiles.
#define MAX_PILLS 20 // Assume a patient takes no more than 20 pills.
/*
   Each row in the following represents one pill. The columns represent
   when the pill will be dispensed. 0 = pill is not dispensed. 1 = pill
   is dispensed.
*/
int dispDayPill[MAX_PILLS][7] = {0}; // Columns represent one day.
int dispHourPill[MAX_PILLS][24] = {0}; // Columns represent one hour.
void inputPillInfo(void); // Walks user through entering prescription.
void addDays(void); // Adds the days for current pill to be dispensed.
int currentPill = 1; // Variable used to index into dispense arrays.
void addHours(void); // Adds the hours at which pill will be dispensed.
int pillCounts[MAX_PILLS][1] = {0}; // Stores how many pills are loaded.
void addPillCount(void); // Adds the number of pills loaded.
//-----------------------------------------------------------------------

//-----------------------------------------------------------------------
// Initialize variables and functions for current time and day.
int day, hour, min;
struct tm* ptr;
time_t t;
```

```
void updateNow(void); // Updates the current time.

// int day = 0, hour = 0, min = 0; // DEMO Demo starts on Sunday at
00:00.
// timeCounter = -15; // DEMO used in demo code
//---------------------------------------------------------------------

//---------------------------------------------------------------------
/*
Initialize functions and variables for checking for dispense and
monitoring if a patient has retrieved their pills.
*/
void checkDispense(void); // Checks if any pills need to be dispensed.
/*
Time to wait for the patient to take pills with each notification. In
actual
use this time should be 5 minutes.
*/
#define WAIT 300
// #define WAIT 5 // DEMO value
int pillFlag = 0; // Variable to monitor if pills have been taken.
void monitorPillStatus(void); // Changes notification based on pill
status.
void checkForRefill(); // Alerts caregiver if refill is needed.
//---------------------------------------------------------------------

//---------------------------------------------------------------------
// Initialize variables and functions for changing prescription.
void checkForUpdates(); // Allows user to update prescriptions.
#define WAIT2 15 // Wait time for user input. Shorter for demo.
//#define WAIT2 0.5 // DEMO  value
//---------------------------------------------------------------------

//---------------------------------------------------------------------
// Displays values in prescription arrays to show update actions.
void printCheck();
//---------------------------------------------------------------------

//---------------------------------------------------------------------
//---------------------------------------------------------------------
int main(int argc, char *argv)
{
      /*
      On first startup, the user will enter the current number of pills
      in the patient's prescription. Then, they will be guided through
      entering the details for each pill.
      */
      int count; // The number of pills to be entered at startup.
      printf("Enter number of pills: ");
```

```
        scanf(" %d", &count);
        getchar(); // Clear the input line.

        /*
        For loop calls the inputPillInfo() function to allow user to enter
        details for each pill. The currentPill is incremented each time.
        */
        for (int i = 0; i < count; i++)
        {
              inputPillInfo();
              currentPill++;
        }
        printf("\nFinished    initial    prescription    input.    Clearing
screen.\n\n");
        printf("Current input: \n");
        printCheck();
        printf("\n");
        Sleep(5000);
        system("clear");

        // The while loop will run indefinitely.
        while(1)
        {
              // Update the current time.
              updateNow();
              /*
              Pills are dispensed on the hour mark. When the current
              minute is 0, check if any pills need to be dispensed.
              */
              if (min == 0)
              {
                    printf("Checking for dispense. \n");
                    printf("Current Day: %d  Time: %d:", day+1, hour);
                    if (min < 10)
                    {
                          printf("0%d\n", min);
                    }
                    else
                    {
                          printf("%d\n", min);
                    }
                    /*
                    Check dispense output what pills are being dispensed at
the
                    current time. It will also update the pill counts.
                    */
                    checkDispense();
                    //  The  counts  are  checked  to  determine  if  refill  is
needed.
```

```
                        checkForRefill();
                }
                /*
                The user can make changes to the prescription as long as the
                time is well before the hour mark. This will ensure that no
                pill dispenses are missed.
                */
                else if (min < 55)
                {
                        system("clear");
                        // checkForUpdates() allows the user to make changes.
                        checkForUpdates();
                }
                // An idle state is included to ensure no dispenses are
missed.
                else
                {
                        system("clear");
                        printf("Idle state. No updates allowed.\n");
                        //Sleep(10); // For demo, sleep for 1 second then check
again.
                }
        }

        return 0;
}
//----------------------------------------------------------------------
//----------------------------------------------------------------------

void printCheck()
{
        // print check
        for (int j = 0; j<5; j++)
        {
                for (int i = 0; i<7; i++)
                {
                        printf("%d", dispDayPill[j][i]);
                }
                printf("\t");

                for (int i = 0; i<24; i++)
                {
                        printf("%d", dispHourPill[j][i]);
                }
                printf("\t%d", pillCounts[j][1]);
                printf("\n");
        }
}
```

```
//----------------------------------------------------------------------
/*
After pills have been dispensed, this function can be run to determine
if any tracks need to be refilled. If the pill count is less than 10
for any pill, the caregiver will be notified.
*/
void checkForRefill()
{
	int flag = 0; // Tracks if any pills need to be refilled.
	int refill[MAX_PILLS] = {0}; // Updates if pill needs refill.
	for (int i = 0; i<MAX_PILLS; i++) // Check each track.
	{
		/*
		Check if this track dispenses pills. If track is not used
then
		it will not signal a refill needed.
		*/
		for (int j = 0; j < 7; j++)
		{
			if (dispDayPill[i][j] == 1) // A 1 indicates track is
used.
			{
				// If the count is low, update the refill array.
				if (pillCounts[i][1] < 10)
				{
					flag = 1; // Change flag if refill is
needed.
					refill[i] = 1;
				}
			}
		}
	}
	// If there are pills to be refilled, display message.
	if (flag == 1)
	{
		printf("\nThe following tracks have less than 10 pills: ");
		for (int i = 0; i<MAX_PILLS; i++)
		{
			if (refill[i] == 1)
			{
				printf("%d  ", i+1);
			}
		}
		printf("\nRefill suggested. Notification sent to
caregiver.\n");
		Sleep(5000);
	}
}
//----------------------------------------------------------------------
```

```
//----------------------------------------------------------------------
/*
This function allows the user to enter in an action to update the
patient's prescription. This function was adapted from
https://stackoverflow.com/questions/7226603/timeout-function
to have desired input and output types.
*/
void checkForUpdates()
{
    char            input[20] = {0};
    fd_set          input_set;
    struct timeval  timeout;
    int             ready_for_reading = 0;
    int             read_bytes = 0;

    /* Empty the FD Set */
    FD_ZERO(&input_set );
    /* Listen to the input descriptor */
    FD_SET(STDIN_FILENO, &input_set);

    /* Waiting for some seconds */
    timeout.tv_sec = WAIT2;    // WAIT seconds
    timeout.tv_usec = 0;    // 0 milliseconds

    /*
      The following defines allowed inputs and their actions for
      changing the prescription.
      */
    printf("Enter action: \n");
      printf("\tU: Update existing pill\n");
      printf("\tR: Refill existing pill\n");
      printf("\tN: Add new pill\n");
      printf("\tD: Delete existing pill\n");

    // Listening for input stream for any activity.
    ready_for_reading = select(1, &input_set, NULL, NULL, &timeout);
    /*
      Here, first parameter is number of FDs in the set, second is
      FD set for reading, third is the FD set in which any write
      activity needs to be updated, which is not required in this
      case. Fourth is timeout.
     */

    if (ready_for_reading == -1)
      {
        /* Some error has occurred in input */
        printf("Unable to read your input\n");
    }
```

```
        if (ready_for_reading)
          {
            read_bytes = read(0, input, 19);
            if(input[read_bytes-1]=='\n')
                 {
            --read_bytes;
            input[read_bytes]='\0';
            }
            if(read_bytes==0)
                 {
                printf("You just hit enter\n");
            }
                 else
                 {
                       /*
                       If an appropriate input is entered, the user will be
                       guided through the needed update actions.
                       */
                       if (input[0] == 'u' || input[0] == 'U')
                       {
                             printf("Update Existing:\n");
                             printf("Enter pill (track number) to update: ");
                             int temp;
                             scanf("%d", &temp);
                             printf("\n");
                             getchar(); // Clear the input line.
                             printf("Before update: \n");
                             printCheck();
                             printf("\n");
                             currentPill  =  temp;  //  Update  the  pill  to  be
edited.
                             inputPillInfo();  //  Call  the  input  function  to
update.
                       }
                       else if (input[0] == 'r' || input[0] == 'R')
                       {
                             printf("Refill Existing:\n");
                             printf("Enter pill (track number) to refill: ");
                             int temp;
                             scanf("%d", &temp);
                             getchar(); // Clear the input line.
                             printf("Enter new pill count: ");
                             int temp2;
                             scanf("%d", &temp2);
                             getchar(); // Clear the input line.
                             printf("Before update: \n");
                             printCheck();
                             printf("\n");
```

```
                        pillCounts[temp-1][1]  =  temp2;  //  Update  the
    pillCount.
                    }
                    else if (input[0] == 'n' || input[0] == 'N')
                    {
                        printf("Add New Pill:\n");
                        // Determine next open track.
                        int i = 0;
                        while (pillCounts[i][1] > 0)
                        {
                            i++;
                        }
                        /*
                        When  the  loop  breaks,  i  has  the  index  of  the
    first
                        unused track. This will become the current pill.
                        */
                        currentPill = i+1;
                        printf("Before update: \n");
                        printCheck();
                        printf("\n");
                        inputPillInfo();  //  Call  the  input  function  to
    update.
                    }
                    else if (input[0] == 'd' || input[0] == 'D')
                    {
                        printf("Delete Existing:\n");
                        printf("Enter pill (track number) to delete: ");
                        int temp;
                        scanf("%d", &temp);
                        printf("Before update: \n");
                        printCheck();
                        printf("\n");
                        // Clear the arrays at the index chosen.
                        for (int i = 0; i<7; i++)
                        {
                            dispDayPill[temp-1][i] = 0;
                        }
                        for (int i = 0; i<24; i++)
                        {
                            dispHourPill[temp-1][i] = 0;
                        }
                        pillCounts[temp-1][1] = 0;
                    }
                    else
                    {
                        printf("Not a correct action.\n");
                    }
                    printf("After update: \n");
                    printCheck();
                    printf("\n");
                    Sleep(5000);
                    system("clear");
            }
        }
          else
          {
                // If not actions are entered print a new line.
            printf("\n");
        }
}
//----------------------------------------------------------------------

//----------------------------------------------------------------------
/*
This function is called to determine if the patient has taken their
pills. Pills being taken is simulated by inputting anything and
pressing enter. There are 3 levels of notification in an attempt to
give the patient an opportunity to take their medication on their own.

This function is called from the checkDispense() function.

The same timeout function from the checkForUpdates() function is again
modified here to allow for the appropriate input and output.
*/
void monitorPillStatus ()
{
    char            input[20] = {0};
    fd_set          input_set;
    struct timeval  timeout;
    int             ready_for_reading = 0;
    int             read_bytes = 0;

    /* Empty the FD Set */
    FD_ZERO(&input_set );
    /* Listen to the input descriptor */
    FD_SET(STDIN_FILENO, &input_set);

    /* Waiting for some seconds */
    timeout.tv_sec = WAIT;    // WAIT seconds
    timeout.tv_usec = 0;    // 0 milliseconds

    // Prompt for an input when pills are taken.
    printf("Enter anything when pills are taken.\n");
    /* Listening for input stream for any activity */
    ready_for_reading = select(1, &input_set, NULL, NULL, &timeout);
    /*
      Here, first parameter is number of FDs in the set, second is
      FD set for reading, third is the FD set in which any write
      activity needs to be updated, which is not required in this
      case. Fourth is timeout.
     */

    if (ready_for_reading == -1)
      {
        /* Some error has occurred in input */
        printf("Unable to read your input\n");
    }

    if (ready_for_reading)
      {
        read_bytes = read(0, input, 19);
        if(input[read_bytes-1]=='\n')
             {
        --read_bytes;
        input[read_bytes]='\0';
        }
             /*
             If pills have been taken, the message is printed and the
pillFlag
             is updated.
             */
        if(read_bytes==0)
             {
            printf("Pills taken.\n");
                   pillFlag = 1;
        }
             else
             {
            printf("Pills taken.\n");
                   pillFlag = 1;
        }
    }
      // If pills are not taken, the message is printed.
      else
      {
             //printf("Pills not taken.\n");
             pillFlag = 0;
    }
      return;
}
//---------------------------------------------------------------------

//---------------------------------------------------------------------
/*
This function will be called when the minute value is 0. It will
determine if any pills need to be dispensed. If pill are dispensed
```

```
the monitorPillStatus function is called to determine if the patient
has taken their pills. Depending on the pillFlag status, the
notification type will be updated.
*/
void checkDispense(void)
{
	int flag = 0; // Flag updates if there are pills to dispense.
	int dispenseNow[MAX_PILLS] = {0}; // Reset the dispense array.
	for (int i = 0; i < MAX_PILLS; i++) // Checks each track.
	{
		if (dispDayPill[i][day] == 1) // Checks the day this day.
		{
			if (dispHourPill[i][hour] == 1) // Checks the hour.
			{
				dispenseNow[i] = 1; // Updates dispense array.
				// Update pill count.
				pillCounts[i][1] = pillCounts[i][1] - 1;
				flag = 1; // Update flag.
			}
		}
	}
	if (flag ==1) // Print which pills are dispensed if any.
	{
		printf("\nDispensing these pills: ");
		for (int i = 0; i < MAX_PILLS; i++)
		{
			if (dispenseNow[i] == 1)
			{
				printf("%d    ", i+1);
			}
		}
		printf("\n");

		// Monitor pill status and update notification as needed.
		pillFlag = 0; // Reset pillFlag.
		// Display first notification.
		printf("\nNotification 1: LED flashing every 10 seconds.\n");
		// Monitor pill status and wait for input.
		monitorPillStatus();
		//  If  pills  are  not  taken  after  notification  1,  update
notification.
		if (pillFlag == 0)
		{
			printf("\nPills not taken in 5 minutes.\n");
			// Display second notification.
			printf("Notification  2:  LED  on,  sound  buzzer  every  15
seconds.\n");
			// Monitor pill status and wait for input.
			monitorPillStatus();
```

```
            }
            // If pills are not taken after notification 2, update
notification.
            if (pillFlag == 0)
            {
                  printf("\nPills not taken in 10 minutes.\n");
                  // Display third notification
                  printf("Notification 3: Alert sent to caregiver. LED
on, sound buzzer every 15 seconds.\n");
                  // Monitor pill status and wait for input.
                  monitorPillStatus();
            }
            /*
            If pills are not taken after notification 3, assume caregiver
            will notify patient of taking their pills. Allow dispenser to
            continue monitoring for future dispenses.
            */
            if (pillFlag == 0)
            {
                  printf("\nPills not taken in 15 minutes. Caregiver will
ensure pills are taken.\n");
            }
            printf("\nMonitor for next dispense.\n");
            Sleep(2000);
      }
      else
      {
            printf("No pills to dispense. Dispenser locked. \n");
            Sleep(1000);
      }
}
//----------------------------------------------------------------------

//----------------------------------------------------------------------
// Updates the current time and day.
void updateNow(void) {

      // These variables are used in the actual simulation.
      t = time(NULL);
    ptr = localtime(&t);
      day = ptr->tm_wday;
      hour = ptr->tm_hour;
      min = ptr->tm_min;

      /*
      // These variable are used in the demo simulation.
      timeCounter = timeCounter + 15; // Counter initially starts at -1.

      There are 1440 minutes in a day. In the demo simulation, a counter
```

```
	will be used to go through each minute. When 60 minutes passes,
	the hour will be updated and the minutes reset. When 24 hours
passes,
	the day will be updated and the hours reset. After 7 days (1 week),
	the day will be reset.

	if (timeCounter == 1440) // 1440 minutes in a day.
	{
		day++; // Increase the day if 1440 minutes has gone by.
		if (day == 7) // Reset the week if 7 days have gone by.
		{
			day = 0;
		}
		hour = 0; // Reset the hour.
		min = 0; // Reset the minute.
		timeCounter = 0; // Reset the counter for the next day.
	}
	else
	{
		if (timeCounter%60 == 0) // If 60 minutes has gone by,
update.
		{
			hour++; // Update the hour.
			min = 0; // Update the minute.
		}
		else
		{
			min = timeCounter%60; // Set current minute.
		}
	}
	*/
}
//----------------------------------------------------------------------

//----------------------------------------------------------------------
/*
This function is used to input pill info. Before calling this
function, the currentPill needs to be updated.On startup, an initial
loop can be used to input sequentially.
*/
void inputPillInfo(void)
{
	addDays(); // Add days current pill is taken.
	addHours(); // Add hours current pill is taken each day.
	addPillCount(); // Add the count of current pill filled.
	printf("\n\n");
	return;
}
//----------------------------------------------------------------------
```

```
//---------------------------------------------------------------------
// Asks the user to enter the number of pills loaded.
void addPillCount(void)
{
      // Prompt for input.
      int tempCount;
      printf("Enter the number of pills loaded: ");
      scanf("%d", &tempCount);
      getchar(); // Clear the input line.
      // Store input in current pill index.
      pillCounts[currentPill-1][1] = tempCount;
}
//---------------------------------------------------------------------

//---------------------------------------------------------------------
/*
Asks the user to enter the hours the current pill needs to be dispensed
every day. Updates the dispHourPill row corresponding to the current
pill.
*/
void addHours(void)
{
      // Prompt for input. Allows multiple inputs at once.
      printf("Enter the hours to dispense this pill each day separated by
spaces.");
      printf("\nUse 24 hour clock time (0-23). End with an 'x'.: ");
      int i = 0, tempHours[24] = {0};

      while(scanf("%d", &tempHours[i]))
      {
            i++;
      }
      getchar(); // clear the input line

      /*
      Go through the input and update the corresponding index in a temp
      updateHours[] array.
      */
      int j = 0, updateHours[24] = {0};
      i = 0;

      while(tempHours[i] > 0) // Go through each value of the input.
      {
            /*
            If the current input value is larger than the j index,
            increase j. j corresponds to the index in the hours array.
The
```

```
            value in tempHours corresponds to the actual hour to dispense
at.
            */
            while(tempHours[i] > j)
            {
                  j++;
            }
            updateHours[j] = 1;
            j++;
            i++;
      }
      // Update the actual dispenseHourPill array.
      for (i = 0; i<24; i++)
      {
            dispHourPill[currentPill-1][i] = updateHours[i];
      }

      return;
}
//----------------------------------------------------------------------

//----------------------------------------------------------------------
/*
Asks the user to enter the days the current pill needs to be dispensed.
Updates the dispDayPill row corresponding to the current pill.
*/
void addDays(void)
{
      // Print instructions for entering the days to dispense.
      printf("Current pill (track #): %d\n", currentPill);
      printf("Enter the days this pill will be dispensed:\n");
      printf("\t 1: Sunday\n");
      printf("\t 2: Monday\n");
      printf("\t 3: Tuesday\n");
      printf("\t 4: Wednesday\n");
      printf("\t 5: Thursday\n");
      printf("\t 6: Friday\n");
      printf("\t 7: Saturday\n");

      // Prompt for input.
      printf("Enter the numbers separated by spaces. End with an 'x'.:
");

      int i = 0, tempDays[7] = {0};

      // Store input in a temp array.
      while(scanf("%d", &tempDays[i]))
      {
            i++;
      }
      getchar(); // Clear the input line.

      int j = 1, updateDays[7] = {0};
      i = 0;
      // Go through each value in the temp array.
      while(tempDays[i] > 0)
      {
            /*
            If the current input value is larger than the j index,
            increase j. j corresponds to the index in the days array. The
            value in tempDays corresponds to the actual day to dispense.
            */
            while(tempDays[i] > j)
            {
                  j++;
            }
            updateDays[j-1] = 1;
            j++;
            i++;
      }
      // Update the actual dispenseHourPill array.
      for (i = 0; i<7; i++)
      {
            dispDayPill[currentPill-1][i] = updateDays[i];
      }

      return;
}
//--------------------------------------------------------------------
```

References

[1] Westbrook, J. I., Woods, A., Rob, M. I., Dunsmuir, W. T., & Day, R. O. (2010). Association of interruptions with an increased risk and severity of medication administration errors. Archives of Internal medicine, 170(8), 683-690

[2] Arlt, S., Lindner, R., Rösler, A., & von Renteln-Kruse, W. (2008). Adherence to medication in patients with dementia. Drugs & aging, 25(12), 1033-1047.

[3] McDonald, H. P., Garg, A. X., & Haynes, R. B. (2002). Interventions to enhance patient adherence to medication prescriptions: scientific review. Jama, 288(22), 2868-2879.

[4] The Autonomous Pill Dispenser: Mechanizing the Delivery of Tablet Medication Shaantam Chawla Mechatronics Research Laboratory Academy for Technology and Computer Science Hackensack, NJ 07601 USA.

[5] Intelligent Pillbox: Automatic And Programmable Assistive Technology Device Juan Marcelo Parra 1, Wilson Valdez \ Andrea Guevara\ Priscila Cedillo2, Jose Ortiz-Segarra 3 University Of Cuenca