CS19003 Programming and Data Structures Lab

Assignment Set 2 **March 21, 2023**

INSTRUCTIONS

1. There are three assignments in this Lab and a non-gradable practice task. You need to submit each of the three assignments separately. It is advisable to submit each assignment as you complete it, rather than wait for the end to submit everything.
2. Your source program files must be named exactly as indicated (note that names are case sensitive)
3. Please write a header as indicated in the previous assignment

---

1. **[Filename: set2asg1.c]**

Computing Semester Results. The number of grade points earned by a student in a subject with X credits is computed as X times Z, where Z is 10 for Ex grade, 9 for A grade and so on. The SGPA of a student in that semester is the total grade points earned divided by the total credits. The curriculum for the second semester for undergraduate students is given below.

Course No.	Name	Credits
MA10002	Mathematics	4
CY11001	Chemistry	4
CS11001	Prog & Data Sruct.	4
ME10001	Mechanics	4
CS19001	PDS Lab	2
CE13001	Engg. Drawing	3
CY19001	Chemistry Lab.	2

Write a C program, that does the following:

(a) It prompts the user to enter the grades for each of the above subjects and then computes (and prints) the SGPA of that student. This is repeated for 10 students. For convenience, we assume that the user will enter the letter X for the Ex grade.

(b) It prints the average SGPA of the 10 students, the maximum SGPA and the minimum SGPA. It does this without storing the SGPA of the 10 students.

(c) For each subject, it prints the average grade points obtained by the students in that subject. It does this without storing any grades.

(d) It prints the course numbers of the subjects having the most number of Ex grades and the most number of F grades respectively.

2. **[Filename: set2asg2.c]**

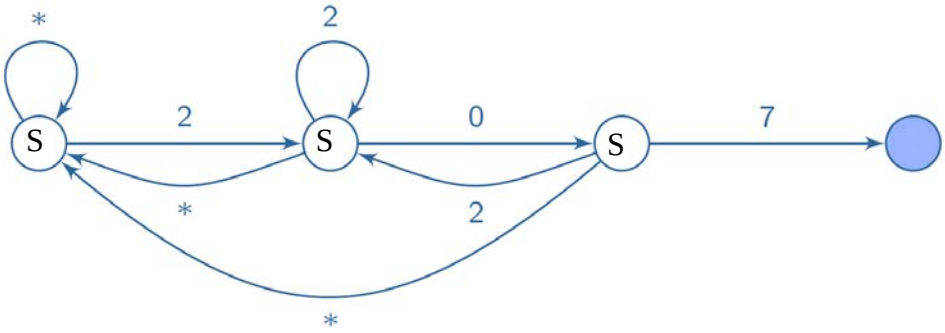
Currency Exchange. In a country currency notes are available in denominations of 500, 100, 50, 20, 10 and 5 rupees respectively. Write a C program which reads any integer, N, and prints the combination having the minimum number of currency notes that add up to N. For example, for N=3785 your program should print:

3785 = 500 X 7 + 100 X 2 + 50 X 1 + 20 X 1 + 10 X 1 + 5 X 1

This indicates that seven 500-rupee notes, two 100-rupee notes and one each of 50, 20, 10 and 5-rupee notes is the minimum number of notes that add up to 3785. If N is not divisible by 5, your program should print that no exchange is possible.

3. **[Filename: set2asg3.c]**

Numerical Door Lock. A door lock opens when the code **207** is pressed in its keypad. The user may press any sequence of keys, numbered 0, 1, …, 9, but the lock does not open until the user presses 2 followed by 0 followed by 7. Since this is a 3-digit code, the lock needs to remember the previous two key presses to determine whether 207 has been pressed in sequence. In programming we often need to model such *states* of the device we wish to program. For example, the figure shown below is a state machine for the numerical door lock with code 207.



The rightmost (blue) state is reached when the previous three key presses are 2-0-7 and the lock is open. State S3 represents the situation where the previous two key presses are 2-0 (and therefore the first two digits of the key has matched). If the next key press is 7, then the lock moves to the blue state and opens. State S2 represents the situation where the previous key press was 2 (and therefore the first digit of the key has matched). From this state it goes to S3 if the next key pressed is 0. S1 represents the state where no part of the key has matched. The labels on the transitions (edges) between the states are the key presses. We use * to denote a wildcard or default transition. From a state we take the default transition when the next key press does not match with any of the other transitions from that state.

Your task is to write a C program that continues to read a sequence of digits, one at a time, until the digits 2, 0, 7 arrive one after the other. When that happens, the program must print: `Lock is open` and terminate.

The following code skeleton is provided as a hint:

```
state = 1;   /* Initial state is S1 */
open = 0;    /* This is the status of the lock. open==0 means lock is closed */
while (open == 0) {
      c = getchar(); /* Read a digit from the keypad */
      switch (state) {
            case 1: /* You need to complete this part */ break;
            case 2: /* You need to complete this part */ break;
            case 3: if (c == '2') state = 2;
                        else if (c == '7') {
                              open = 1;
                              printf("Lock is open \n");
                        }
                        else state = 1;
                        break;
      }
}
```

[Point to ponder: How would things change is the code was 22022 ? This is not part of the assignment, but good to think about.]

4. **[Remark: This is a practice exercise. You will not submit this program.]**

Learning to read from files. Make a copy of the C program from the previous assignment, and modify it so that the sequence of key presses is read from a text file, **set2asg3.txt**. You must create and edit this file first using a text editor like gedit, so that it contains a sequence of digits – one digit in each line. You can read the input from the file using a variant of the following code.

```
FILE fopen(), *fp;
int digit;

/* The following line opens the file and positions the file pointer on the first character */
if ( ( fp = fopen( "set2asg3.txt",  "r" ) ) < 0) {
    printf("Could not open file set2asg3.txt. Quitting program\n"); exit(0);
}
------
------

fscanf(fp, "%d", &digit);          /* Reads the next keypress into digit */

------
------
fclose( fp );     /* Close the file at the very end */
```

It is known that `fscanf()` returns EOF when the program has reached the end of the input file. We can therefore break out of the reading loop using something like the following in place of the above:

```
if (fscanf(fp, "%d", &digit) == EOF) break;
```