# Midterm: LCS circuit

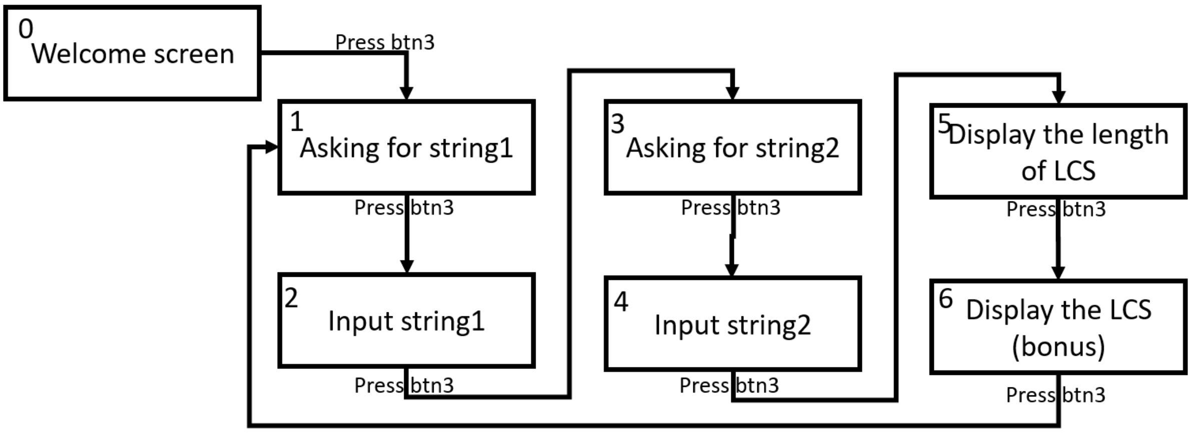
## 考場規則(Regulation)

- Online test exam is individual exam NOT team exam.
- You can take one USB flash disk to store the necessary data and you have to back up to the PC you used before the exam. During the exam, no internet and no outsourcing data.
- The TA will ask you some questions if needed for each online test exam. Your answer will be part of grade.
- Please do NOT copy and do NOT plagiarize. If true by verification, your grade for this exam will be zero. That means if TA finds out the (part) answer you show is copied from others, the exam will be zero.

## 摘要(Abstract)

In this midterm exam, you will design a circuit that could take two input strings from the user through buttons and LCD, then calculate the LCS(Longest common subsequence) between 2 strings. Besides, the 1602-LCD screen will be used to display the strings. The flow chart is shown in Fig 1, and detailed explanation of the circuit would be shown below.

Fig1. FLOW CHART:



## PART A(10%) 1602-LCD DISPLAY (marked as number 0 in Fig 1)

Show the welcome string on the LCD screen, like below:

```
welcome!  0123456
Press btn3 start
```

Note:0123456 is your student id. If your id is longer than 7 digits, showing the last 7 digits

.

## PART B(60%) SELECT INPUT STRING THROUGH BUTTON (marked as number 1~4 in Fig 1)

You will have a circular string “[0-9][A-Z][a-z]”, which means the next character of the last character is the first character. The circular string is displayed on the first row of LCD screen, and you need to use buttons to control it scrolling left or right.

In the second row of the LCD screen, you will use “^” to select the character from the first row and show it on the second.

```
yz0123456789ABCD
LLOWORLD       ︿
```

**Note:You may assume that the user’s input string is less than 16 characters.**

## PART B-1(10%) SHOWING MESSAGE (marked as number 1、3 in Fig 1)

```
press btn 3 to
enter string #k
```

Note: k = 1 in section 1, k = 2 in section 3,.

## PART B-2(30%) SELECT INPUT STRING (marked as number 2、4 in Fig 1)

Users need to use the 4 buttons to control the first row to input the string. The function of each button is listed:

| btn3 | btn2 | btn1 | btn0 |
|---|---|---|---|
| input complete | scrolling left | choosing a character | scrolling right |

The user could use btn2 or btn0 to scroll left and right, respectively, the first row of the LCD screen, once the desired character is on the top of ‘^’, press btn1 to append the character to the back of string. After the whole string was entered, press btn3 to tell the circuit you have completed your string.

For example:

The user wants to enter “Helloworld”, he should do the following things:

1、initial status: The screen would be like :

```
l m n o p q r s t u v w x y z 0
                              ︿
```

input string register:

2、press btn2 to scroll left the until the ‘^’ reach ‘H’

```
2 3 4 5 6 7 8 9 A B C D E F G H
                              ︿
```

input string register:

３、press btn1 to select input character

```
2 3 4 5 6 7 8 9 A B C D E F G H
H                             ︿
```

input string register: “H”

４、press btn2 scroll left until the ‘^’ reach ‘i’

```
P Q R S T U V W X Y Z a b c d e
H                             ︿
```

input string register: “H”

5、press btn1 to select input character

```
P Q R S T U V W X Y Z a b c d e
H e                           ︿
```

input string register: “He”

6、press btn2 scroll left until the ‘^’ reach ‘a’

```
W X Y Z a b c d e f g h i j k l
H e                           ︿
```

input string register: “Hel”

7、press btn1 to select input character

```
WXYZabcdefghijkl
Hel            ︿
```

input string register: “Hel”

.

.

.

8、After enter the whole string, press btn3 to take the input string.

input string register: “Helloworld”

## PART B-3(20%) DISPLAY SELECT INPUT (marked as number 2、4 in Fig 1)

The first 8 characters of the second row of the LCD screen should echo the input string in real time. The LCD screen should follow the rule below:

For example: The user wants to enter the string “Helloworld”

1. The user has not entered any character, yet. The first row should display the circular string. The first 8 characters of second row should display blanks, and the last character should display ‘^’

```
lmnopqrstuvwxyz0
               ︿
```

2. After the user enters some characters (with length < 8), the second row should display the string that the user has entered.

```
VXYZabcdefghijkl
Hel            ︿
```

3. After the user enters some characters (with length “k” >= 8), the second row should scroll left and display the (k-7)-th to the k-th character of the user’s input string.

```
OPQRSTUVWXYZabcd
lloworld       ︿
```

## PART C(30%) Calculate the length of LCS (marked as number 5 in Fig 1)

In this part, you need to implement a dynamic programming algorithm to calculate the length of the longest common subsequence (in hexadecimal) between user's input strings. Then display on the LCD screen.

```
The length of
LCS = 0x6
```

(Algorithm and LCS are given in the reference pages.)

## BONUS (30%) Display LCS(marked as number 6 in Fig 1)

Before testing new strings again, press button 3 to print the longest common subsequence on the LCD screen.

**Note:You may assume that the LCS is less than 16 characters.**

```
The LCS is:
DLAAAB
```

**Your circuit will be tested with different inputs. Your final score will be the average of them.**

# REFERENCE

## Longest common subsequence

The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). It differs from the longest common substring problem: unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences. The longest common subsequence problem is a classic computer science problem, the basis of data comparison programs such as the diff utility, and has applications in computational linguistics and bioinformatics. It is also widely used by revision control systems such as Git for reconciling multiple changes made to a revision-controlled collection of files

Examples:

LCS for input Sequences “ABCDGH” and “AEDFHR”   is “ADH” of length 3.

LCS for input Sequences “AGGTAB” and “GXTXAYB” is “GTAB” of length

FROM:https://en.wikipedia.org/wiki/Longest_common_subsequence_problem

## Here is sample c code for LCS

```
/* Returns length of LCS for X[0..m-1], Y[0..n-1] */
int lcs( char *X, char *Y, int m, int n )
{
   int L[m+1][n+1], LCS[m+1][n+1];
   int i, j;

    /*Following steps build L[m+1][n+1] in bottom up fashion. Note
    that L[i][j] contains length of LCS of X[0..i-1] and Y[0..j-1]
    LCS[i][j] record how to trace the LCS*/

   for (i=0; i<=m; i++){
     for (j=0; j<=n; j++){
       if (i == 0 || j == 0)
         L[i][j] = 0;
       else if (X[i-1] == Y[j-1]){
         L[i][j] = L[i-1][j-1] + 1;
         LCS[i][j] = 0;
       else{
         if (L[i - 1][j] < L[i][j - 1]) {
           L[i][j] = L[i][j - 1];
           LCS[i][j] = 1;// LCS from the left
           }
         else {
           L[i][j] = L[i - 1][j];
           LCS[i][j] = 2;// LCS from the top
         }
       }
     }
   }
/* L[m][n] contains length of LCS for X[0..n-1] and Y[0..m-1] */
   return L[m][n];
}
```

## How to print the LCS?

```
{
   /* The variables here is the same as in last page */
    int len = L[m][n];
    char seq[len+2] = {0};

    while (len > 0) {
        if (LCS[m][n] == 0) {
            len--;
            m--;
            n--;
            seq[len] = X[m];
        }
        else if (LCS[m][n] == 1)
            n--;
        else
            m--;
    }
    printf("%s", seq);
}
```

FROM:https://yungshenglu.github.io/2018/05/15/LongestCommonSubsequence1/

# ASCII Table

| Characters | ASCII code (dec) | Characters | ASCII code (dec) | Characters | ASCII code (dec) |
|---|---|---|---|---|---|
| 0-9 | 48-57 | A-Z | 65-90 | a-z | 97-122 |
| space | 32 | ^ | 94 | | |

# The END