Problem 4 (23 points)

Your goal in this problem is to write a C program, called `strings.c`, that performs some simple string analysis and manipulation. To help you write this program, the relevant part of the ASCII code table is given below. As you can see, the ASCII codes of the upper-case letters **A** to **Z** are in sequence from 65 to 90, respectively. The ASCII codes of the lower-case letters **a** to **z** are also in sequence, from 97 to 122, respectively.

character	A	B	C	···	X	Y	Z	[	\	]	^	_	`	a	b	c	···	x	y	z
ASCII code	65	66	67	···	88	89	90	91	92	93	94	95	96	97	98	99	···	120	121	122

The program `strings.c` first prompts the user to enter a string, consisting of a contiguous sequence of letters. You should assume that the user complies and enters a string consisting of letters (upper-case **A**–**Z** or lower-case **a**–**z** or both) only, without whitespace between them. You should also assume that the user string will have at most **MAXLEN** $- 1 = 127$ characters. The program then further prompts the user to enter a single letter. You should again assume that the user complies, and enters a single character which is either an upper-case letter [**A**–**Z**] or a lower-case letter [**a**–**z**]. This letter may or may not occur in the user string.

After that, the program reports to the user the *length* of the string the user has entered (the total number of characters in it, not including the `'\0'` character). The program also displays the user string with all letters converted to upper case and with all letters converted to lower case. The program then counts the number of times the single character entered by the user appears in the user string (note that this count is *case-sensitive*, e.g. the letters **A** and **a** are considered different). If this number is nonzero, the program reports it to the user, along with the list of positions where this character occurs. If the character does not occur at all in the user string, the program simply reports this to the user.

Finally, the program counts how many *different* letters there are in the user string. For example, if the string is **ThisIsMyFavoriteCourse**, then the total number of characters in the string (its length) is 22, but the number of different characters in this string is 14. Note that this count is *case-insensitive*. For example, in the string **ThisIsMyFavoriteCourse**, the letters **T** and **t** (as well as **I** and **i**) are considered the same. Here is a sample run of the program.

```
/home/userXYZ/ECE15/Final> strings
Enter a string [letters only]: ThisIsMyFavoriteCourse
Enter a letter to find in your string: i
The length of your string is 22.
Your string in upper case: THISISMYFAVORITECOURSE
Your string in lower case: thisismyfavoritecourse
The letter i occurs 2 times in ThisIsMyFavoriteCourse
at the following positions: 3 14
The number of different letters in your string is 14.
```

Provided on the next page is part of the source code that implements the program `strings.c`. This part contains the forward function declarations, and the complete implementation of the function **main()**. Your task is to implement the other functions. Note that in this problem, you are *not* allowed to use ***any*** functions from the C standard library, except for those declared in `<stdio.h>`.

```c
#include <stdio.h>
#define MAXLEN 128

/* Forward function declarations */
int string_length(char str[]);
void toUpper(char input[],char output[]);
void toLower(char input[],char output[]);
int find(char input[],int positions[],char c);
int different(char input[]);

int main()
{
   char c;
   char str[MAXLEN], str2[MAXLEN];
   int i, times_found;
   int positions[MAXLEN];

   printf("Enter a string [letters only]: ");
   scanf("%s",str);
   while (getchar() != '\n');

   printf("Enter a letter to find in your string: ");
   scanf("%c",&c);

   printf("The length of your string is %d.\n",
                                 string_length(str));
   toUpper(str,str2);
   printf("Your string in upper case: %s\n", str2);
   toLower(str,str2);
   printf("Your string in lower case: %s\n", str2);

   times_found = find(str,positions,c);
   if (times_found != 0)
   {
      printf("The letter %c occurs %d time%s in %s.\n",
               c,times_found,(times_found==1) ? "":"s",str);
      printf("At the following positions: ");
      for (i = 0; i < times_found; i++)
              printf("%d ", positions[i]);
      printf("\n");
   }
   else
      printf("The letter %c does not occur in %s.\n",c,str);

   printf("The number of different letters ");
   printf("in your string is %d.\n", different(str));

   return 0;
}
```

a. The function `main()` calls the function `string_length(str)`, which returns the length (the total number of characters not including the `'\0'` character at the end) of the given string `str[]`. Implement this function below.

```
int string_length(char str[]);
{
```

b. The function `main()` also invokes the functions `toUpper()` and `toLower()`. The function `toUpper(input,output)` takes an input string `input[]` and writes this string into the output string `output[]` while converting all lower-case letters [`a`–`z`] to upper-case letters [`A`–`Z`]. The function `toLower(input,output)` does the same thing, except that it converts from upper case to lower case. Implement these functions below.

```
void toUpper(char input[],char output[])
{
```

```
void toLower(char input[],char output[])
{
```

Notes:

- You can assume that the input string contains *only* letters [`A–Z`] and [`a–z`]. If you do not want to assume this, leave all the other (non-letter) characters unchanged by simply copying them from `input[]` to `output[]`.
- Consult the partial ASCII code table given at the beginning of this problem to figure out how to convert from upper case to lower case, and vice versa.
- You can either process the input string `input[]` until encountering the `'\0'` character or call `string_length(input)` to find out in advance how many characters there are in the input string. Note that `output[]` should also terminate with `'\0'`.