DASF004: Basic and Practice in Programming

- Lab 10: Character and String

In this lab ...

- Be familiar with various operations and functions in manipulating characters and strings

- What you need to submit in this lab (Lab #10):
  - Lab Exercise #10 before the end of today (23:59 pm)
  - Assignment #10 by Tuesday 23:59 pm

List of functions that are related to character and/or string manipulations

- You need to know what are the existing functions, and what they are capable of doing
- You do not need to memorise all the specification/detail of each function
  - » You can look up the documentation when you need to use them

The <ctype.h> library

Prototype	Function description
`int isblank( int c );`	Returns a true value if c is a *blank character* that separates words in a line of text and 0 (false) otherwise. [*Note:* This function is not available in Microsoft Visual C++.]
`int isdigit( int c );`	Returns a true value if c is a *digit* and 0 (false) otherwise.
`int isalpha( int c );`	Returns a true value if c is a *letter* and 0 otherwise.
`int isalnum( int c );`	Returns a true value if c is a *digit* or a *letter* and 0 otherwise.
`int isxdigit( int c );`	Returns a true value if c is a *hexadecimal digit character* and 0 otherwise. (See Appendix C for a detailed explanation of binary numbers, octal numbers, decimal numbers and hexadecimal numbers.)
`int islower( int c );`	Returns a true value if c is a *lowercase letter* and 0 otherwise.
`int isupper( int c );`	Returns a true value if c is an *uppercase letter* and 0 otherwise.
`int tolower( int c );`	If c is an *uppercase letter*, `tolower` returns c as a *lowercase letter*. Otherwise, `tolower` returns the argument unchanged.

The <ctype.h> library

Prototype	Function description
`int toupper( int c );`	If `c` is a *lowercase letter*, `toupper` returns `c` as an *uppercase letter*. Otherwise, `toupper` returns the argument unchanged.
`int isspace( int c );`	Returns a true value if `c` is a *whitespace character*—newline (`'\n'`), space (`' '`), form feed (`'\f'`), carriage return (`'\r'`), horizontal tab (`'\t'`) or vertical tab (`'\v'`)—and 0 otherwise.
`int iscntrl( int c );`	Returns a true value if `c` is a *control character* and 0 otherwise.
`int ispunct( int c );`	Returns a true value if `c` is a *printing character other than a space, a digit, or a letter* and returns 0 otherwise.
`int isprint( int c );`	Returns a true value if `c` is a *printing character including a space* and returns 0 otherwise.
`int isgraph( int c );`	Returns a true value if `c` is a *printing character other than a space* and returns 0 otherwise.

The <stdlib.h> library

Function prototype	Function description
`double strtod( const char *nPtr, char **endPtr );`	Converts the string `nPtr` to `double`.
`long strtol( const char *nPtr, char **endPtr, int base );`	Converts the string `nPtr` to `long`.
`unsigned long strtoul( const char *nPtr, char **endPtr, int base );`	Converts the string `nPtr` to `unsigned long`.

The <string.h> library

Function prototype	Function description
`char *strcpy( char *s1, const char *s2 )`	*Copies* string `s2` into array `s1`. The value of `s1` is returned.
`char *strncpy( char *s1, const char *s2, size_t n )`	*Copies at most* `n` *characters* of string `s2` into array `s1`. The value of `s1` is returned.
`char *strcat( char *s1, const char *s2 )`	*Appends* string `s2` to array `s1`. The first character of `s2` *overwrites the terminating null character* of `s1`. The value of `s1` is returned.
`char *strncat( char *s1, const char *s2, size_t n )`	*Appends at most* `n` *characters* of string `s2` to array `s1`. The first character of `s2` *overwrites the terminating null character* of `s1`. The value of `s1` is returned.

The <string.h> library

Function prototype	Function description
`int strcmp( const char *s1, const char *s2 );`	*Compares* the string `s1` with the string `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`, respectively.
`int strncmp( const char *s1, const char *s2, size_t n );`	*Compares up to `n` characters* of the string `s1` with the string `s2`. The function returns `0`, less than `0` or greater than `0` if `s1` is equal to, less than or greater than `s2`, respectively.

Some examples...

Convert a string to all upper case, using a while loop...

```
#include <stdio.h>
#include <ctype.h>
int main(void)
{ char test = “Hello, World!!!”
  for(int i=0;i<sizeof(test)/sizeof(char);i++)
  { test[i] = toupper(test[i]);
  }
  printf(“Output: %s”,test);
}
```

Some examples...

Putting it into a function

```
#include <stdio.h>
#include <ctype.h>
void stringtoupper(char * input, int size)
{ for(int i=0;i<size;i++)
❖     input[i] = toupper(input[i]);
}

int main(void)
{ char test = “Hello, World!!!”
  stringtoupper(test,sizeof(test)/sizeof(char));
  printf(“Output: %s”,test);
}
```

Lab Exercise 10

A website requires users to make a password when registering an account.

Write a program to check if the password made by the user is a valid password based on the following rules:

- The password should contain at least 1 lower case letter [a-z];
- The password should contain at least 1 upper case letter [A-Z];
- The password should contain at least 1 number [0-9];
- The password should contain at least 1 punctuation; and
- Minimum length of the password is 6 characters.

Your program should ask the user to enter the password, and then display:

“This is a valid password” OR “This is NOT a valid password”

Lab 10 Exercise

- Submit it thru icampus before the deadline:
    - Wednesday 23:59 pm

More examples

```
#include <stdio.h>
#include <string.h>

int main ()
{  char str1[12] = "Hello";
   char str2[12] = "World";
   char str3[12];
   int  len ;

   /* copy str1 into str3 */
   strcpy(str3, str1);
   printf("strcpy( str3, str1) :  %s\n", str3 );

   /* concatenates str1 and str2 */
   strcat( str1, str2);
   printf("strcat( str1, str2):   %s\n", str1 );

   /* total lenghth of str1 after concatenation */
   len = strlen(str1);
   printf("strlen(str1) :  %d\n", len );
   return 0;
}
```

strcmp()

```
printf("Enter you name: ");
scanf("%s", name);
if( strcmp( name, "jane" ) == 0 )
 printf("Hello, jane!\n");
```

The <string.h> library

Function prototypes and descriptions

```
char *strchr( const char *s, int c );
```

Locates the first occurrence of character `c` in string `s`. If `c` is found, a pointer to `c` in `s` is returned. Otherwise, a `NULL` pointer is returned.

```
size_t strcspn( const char *s1, const char *s2 );
```

Determines and returns the length of the initial segment of string `s1` consisting of characters *not* contained in string `s2`.

```
size_t strspn( const char *s1, const char *s2 );
```

Determines and returns the length of the initial segment of string `s1` consisting *only* of characters contained in string `s2`.

```
char *strpbrk( const char *s1, const char *s2 );
```

Locates the first occurrence in string `s1` of any character in string `s2`. If a character from string `s2` is found, a pointer to the character in string `s1` is returned. Otherwise, a `NULL` pointer is returned.

The <string.h> library

Function prototypes and descriptions

```
char *strrchr( const char *s, int c );
```

Locates the last occurrence of `c` in string `s`. If `c` is found, a pointer to `c` in string `s` is returned. Otherwise, a `NULL` pointer is returned.

```
char *strstr( const char *s1, const char *s2 );
```

Locates the first occurrence in string `s1` of string `s2`. If the string is found, a pointer to the string in `s1` is returned. Otherwise, a `NULL` pointer is returned.

```
char *strtok( char *s1, const char *s2 );
```

A sequence of calls to `strtok` breaks string `s1` into *tokens*—logical pieces such as words in a line of text—separated by characters contained in string `s2`. The first call contains `s1` as the first argument, and subsequent calls to continue tokenizing the same string contain `NULL` as the first argument. A pointer to the current token is returned by each call. If there are no more tokens when the function is called, `NULL` is returned.

strspn()

```
#include <stdio.h>
#include <string.h>

int main ()
{
  int i;
  char strtext[] = "129th";
  char cset[] = "1234567890";

  i = strspn (strtext,cset);
  printf ("The initial number has %d digits.\n",i);
  return 0;
}
```

» Output:

```
The initial number has 3 digits.
```

strpbrk()

```
#include <stdio.h>
#include <string.h>

int main ()
{ char str[] = "This is a sample string";  // This is a sample string
  char key[] = "aeiou";
  char * pch;
  printf ("Vowels in '%s': ",str);
  pch = strpbrk (str, key);
  while (pch != NULL)
  { printf ("%c " , *pch);
    pch = strpbrk (pch+1,key);
  }
  printf ("\n");
  return 0;
}
```

Output:

```
Vowels in This is a sample string: i i a a e i
```

strtok()

```
#include <stdio.h>
#include <string.h>

int main()
{ char message[] = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.";
  char* word;

  /* get the first word from the message, seperated by space character */
  word = strtok(message, " ");
  printf("1st word: %s\n", word);

  /* get the second word from the message, NULL must be
   * used to get tokens from the previous string now */
  word = strtok(NULL, " ");
  printf("2nd word: %s\n", word);

  /* the following loop gets the rest of the words until end of the message */
  while ((word = strtok(NULL, " ")) != NULL)
   printf("Next: %s\n", word);

  return 0;
}
```

Lab Assignment #10:

- Write two **functions** to count:

(1) the number of punctuations in the string, and
(2) the number of words in the string.

- You may use the `ispunct()` function to implement the punctuation counting.
- You may assume that each word is always either followed by a space or a punctuation and a space.
  - i.e. counting the space, then calculate the number of words.
- A code segment with 3 testing string is provided to you in the code for testing purpose. Your 2 functions should be working with all string.
- You need to implement the function in the code segment provided to you. The expected result of the program is also provide to you.

Lab Assignment #10:

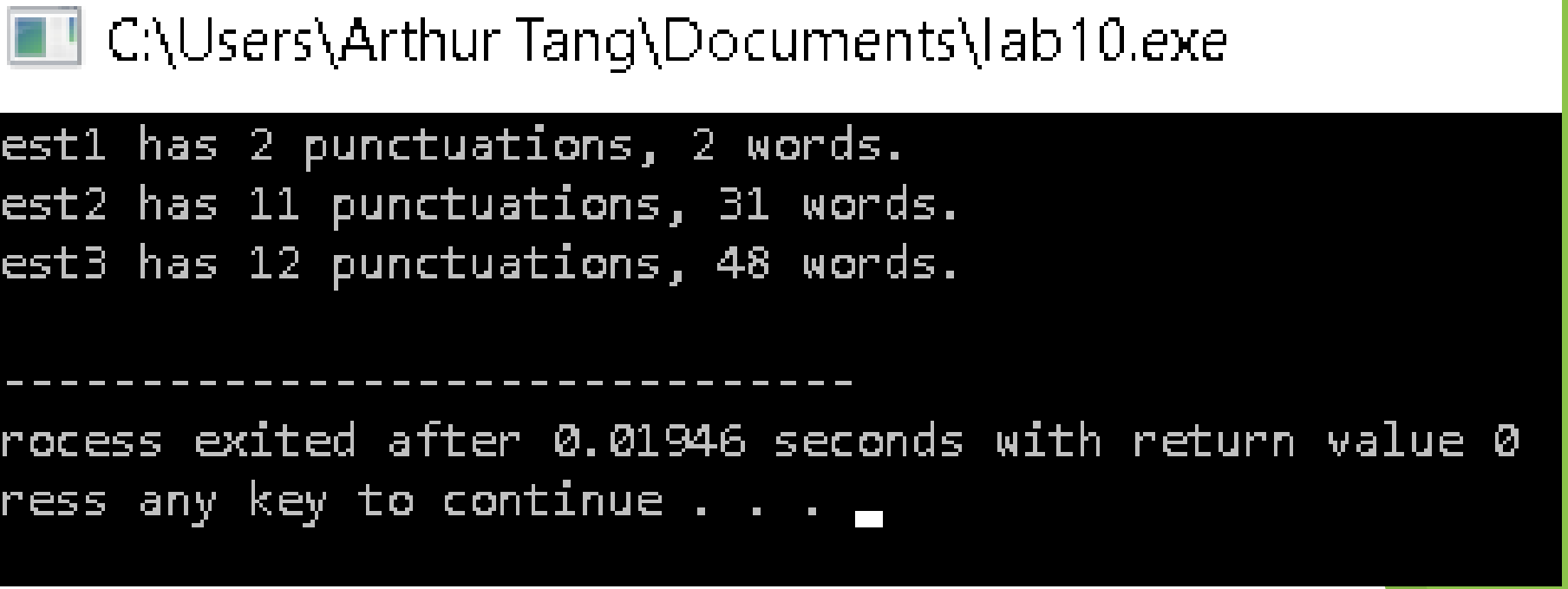
```
#include <stdio.h>
#include <ctype.h>
#include <string.h>

int WordCount(char * input, int size)
{ // You need to complete this function!!!
}

int PunctuationCount(char * input, int size)
{  // You need to complete this function!!!
}

int main(void)
{ char test1[] = "Hello, World!";
  char test2[] = "She sell sea shell on a sea shore. The shells she sells are sea-shells. I'm sure? For if she
       sells sea-shells on the sea-shore, then I'm sure she sells sea-shore shells!";
  char test3[] = "VISION 2020 embodies every SKKU student's dream, will, and destiny to make SKKU a global
       leading university. To successfully establish VISION 2020, SKKU will pursue 'The 5 Core Strategies' and '5
       Divisional Strategies'. The Strategic Tasks for 5 Major Areas are the general tasks that influence SKKU's
       competitiveness.";

  printf("test1 has %d punctuations, %d words.\n", PunctuationCount(test1,sizeof(test1)/sizeof(char)),
         WordCount(test1,sizeof(test2)/sizeof(char)));
  printf("test2 has %d punctuations, %d words.\n", PunctuationCount(test2,sizeof(test2)/sizeof(char)),
         WordCount(test2,sizeof(test2)/sizeof(char)));
  printf("test3 has %d punctuations, %d words.\n", PunctuationCount(test3,sizeof(test3)/sizeof(char)),
         WordCount(test3,sizeof(test2)/sizeof(char)));
  return 0;
}
```


C:\Users\Arthur Tang\Documents\lab10.exe
test1 has 2 punctuations, 2 words.
test2 has 11 punctuations, 31 words.
test3 has 12 punctuations, 48 words.

Process exited after 0.01946 seconds with return value 0
Press any key to continue . . .

Lab Assignment #10: String Operations

Submit your source code on iCampus before Tuesday 11:59 pm