Name: 陳柏瑜

Student ID: 41047054S
Department: Computer Science and Information Engineering 114
Computer Programming II HW01 README

HW0101 My String Library

In this question, I implement 7 string searching functions on my own. Functions implemented are as the following.

1. char *mystrchr(const char *s, int c);
2. char *mystrrchr(const char *s, int c);
3. size_t mystrspn(const char *s, const char *accept);
4. size_t mystrcspn(const char *s, const char *reject);
5. char *mystrpbrk(const char *s, const char *accept);
6. char *mystrstr(const char *haystack , const char *needle);
7. char *mystrtok(char *str , const char *delim);

These functions works the same as the standard C library defined.

Note that you should prepare your own hw0101.c file. I only provide mystring.h header, mystring.c file, and Makefile.

HW0102 String Insertion

Here is the function allows you do string insertion. The function definition is as the following line.

```
int32_t strinsert( char **ppResult , const char *pStr1 , int32_t location , const char *pStr2 );
```

To use this function, you have to prepare an origin string Str1 and a inserting string str2. Then put the pointer of the above 2 string, pStr1 and pStr2 respectively. Also, the location where you wanna insert. Finally, prepare a pointer ppResult, I will put the result into the pointer

with allocated memory.

If there is any error input, the function will return -1, return 0 otherwise.

For the error input, here are some examples.

1. If the number of location is larger than the length of Str1 or it's smaller than 0, it prints an error message "Overflow!" and terminate the function.
2. If either ppResult, pStr1, or pStr2 is NULL pointer, it prints an error message "NULL input!" and terminate the function.

Note that you should prepare your own hw0102.c file. I only provide insert.h header, insert.c file, and Makefile.

HW0103 Abacus

HW0103 is an implementation of abacus. I provide the following 6 functions.

1. `sAbacus * abacus_init( void );`
2. `void abacus_free( sAbacus * );`
3. `int32_t abacus_set( sAbacus *pA , char *pStr );`
4. `int32_t abacus_add( sAbacus *pA , sAbacus b, sAbacus c );`
5. `int32_t abacus_del( sAbacus *pA , sAbacus b, sAbacus c );`
6. `int32_t abacus_print( sAbacus a );`

The implementation is built based on the following structure.

```
typedef struct _sAbacus
{
   uint8_t number; // 0-255
   uint8_t *pUpperRod; // Each element is in {0,1}
   uint8_t *pLowerRod; // Each element is in {0,1,2,3,4}
}sAbacus;
```

The initial function allocate a block of memory, fill 0, and return back.

The free function just free the structure memory.

The set function will set the value of strure pA the correct value with given string pStr.

The add and del functions will conduct addition and subtraction with the given b and c structure, then store the result into a structure.

Last, the print function will print out the structure with color!

For the above functions, they will return 0 if conducting correctly, but return -1 when error occurs. Here are some error cases.

1. NULL input
2. Length of pStr is larger than 255
3. The arithmetic result is less than zero

The feature of these functions may be as following.

1. The print function prints the abacus with color.
2. The deletion functions will delete the beginning zero if this issue occurs. For example, B structure is 1000, and C structure is 999, it will print a result in 1 instead of 0001.

Last, The addition and subtraction function use additional linked list to implement for easily implementation to handle the carry issue. That's why there is an extra header called linkedlist.h. But is has nothing to do with the result. I store the value back to the abacus structure.

Note that you should prepare your own hw0103.c file. I only provide abacus.h header, abacus.c file, and Makefile.

HW0104 JSON Reader

Here is a JSON parser. Just input a JSON string and the key you want to find in the JSON string. I will show you the value of that key. However, I haven't developed complete functions, which means there are some constrains as follows.

First, I assume the JSON string matches the format of the real JSON

string. I didn't do any invalid check on it.
Secondly, after input the JSON string, the program will ask you to input either 0 to terminate the program or 1 to get the value. **Please input only a number and a newline character**. Of course, you can input like 2, 3, etc. But you'll get an error message. And if your input size is larger than two, the program may run with problem.
Thirdly, the program must have problem if you try to find the nonexistent key. So **don't type any nonexistent key when you search.**

HW0105 Potato Ball

In this problem, I implement a program that simulate the scenario based on the problem description. Here, I implement 10 functions as below.

1. `void print_potato(const struct Potato *this);`
2. `void print_magical_potato(const struct Potato *this);`
3. `void potato_dtor(struct Potato **obj);`
4. `void magical_potato_dtor(struct Potato **obj);`
5. `void print_potato_ball(const struct PotatoBall *this);`
6. `void print_magical_potato_ball(const struct PotatoBall *this);`
7. `void potato_ball_dtor(struct PotatoBall **obj);`
8. `void magical_potato_ball_dtor(struct PotatoBall **obj);`
9. `void init_smallten(struct PotatoProducer *obj);`
10. `void init_subarya(struct PotatoProducer *obj);`

Frankly speaking, there are two types of function, one for original potato and the other for magical potato. For each type, there are 5 functions with different application. For instance, producer, fry, dtor, and two print functions.

Note that you should prepare your own hw0105.c file. I only provide oop.h header, oop.c file, and Makefile.

HW0106 (BONUS) wchar_t

Please see hw0106.pdf file.