BIL 105E – Introduction to Scientific and Engineering Computing (C)

Spring 2015-2016

Homework 3

Assignment Date: **19.4.2016**
Due Date: **29.4.2016, 23.00**
Duration: Ten Days

IMPORTANT:

- Don't use or get inspired by any lines of code from any other sources (friends, Internet, etc). **Any similarity, which is beyond reasonable, will be accepted as cheating!**
- Name your program as **student_id.c** and don't forget to test it on your ITU account before submission by using ssh client. Any code that can't be compiled will not be evaluated.
- Please just use the subjects **shown in the class**. Don't use any other statements or data structures.
- You must submit a report based on the format given on Ninova course page.
- Ninova course page must be used for submissions. Late submissions will not be accepted!

The aim of this homework is to implement some character sequence (i.e., string) operations by using pointers and functions. A menu is presented to the user and the selected operation is executed on a character sequence.

At the beginning, the character sequence is empty and no operations can be done on the empty character sequence. The menu includes an option which allows the user to enter a character sequence from the command line. When the user enters a character sequence, it is set as the **current character sequence (ccs)** and the operations can be done on it. However, the operations **must not affect** the original character sequence entered by the user, if not explicitly stated in the function definitions below. That is, when the user selects another operation without entering a new character sequence, the operation is executed on the lastly entered character sequence by the user (i.e., **ccs**).

In this homework, you must use dynamic memory allocation to store your character sequences. You can assume that, user can't enter a character sequence which contains more than 80 characters. When the user wants to enter a character sequence, you allocate a 80-characters memory space dynamically. After getting the input from the user, you re-allocate the memory space which **exactly** fits to your **ccs**. (Be careful that the last element of the character sequence must be '\0' character.) You can then execute operations on the **ccs**.

Since the ccs will not be changed by the character sequence operations, (remember that, **ccs** can only be changed by entering a new character sequence), you have to allocate a new memory space dynamically to store the selected operation's result.

You are only allowed to traverse the character sequences with pointers, **not with square brackets**.

In this homework your program will implement the following steps:

(1) Print the following menu that shows character sequence operations and read the user's operation choice.

```
0: Exit the program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace
```

Implement the following function to print the menu:

```
int user_menu(void);
```

the function prints the menu and returns back the user's operation choice.

(2) If the user enters 0 (i.e., user_menu() function call returns 0), then exit the program.

(3) Else if **ccs** is NULL and user enters an operation other than (1), warn the user that (s)he must first enter a character sequence to operate on.

Then, go to step (1)

(4) Else if the operation selection is (1), then ask for the **ccs:**

```
Enter CCS:
```

Note that you have to allocate memory space **JUST ENOUGH** to store entered character sequence, as described above. Implement the following function to **ask** and **set** the **ccs**:

```
int set_ccs(char **ccs)
```

The function returns the number of bytes that is allocated to store **ccs**. If the allocation is unsuccessful, then it returns -1.

After setting the **ccs**, go to step (1).

(5) Else if the operation selection is (2), then ask for the beginning and end character indexes to get the substring:

Enter the BEGIN INDEX and END INDEX numbers:

Implement the following function prototype for this operation:

char *sub_string (char *ccs, int begin_index, int end_index)

Note that **begin_index** and **end_index** are both INCLUSIVE and indexing starts from 0. That is, the characters at locations **begin_index** and **end_index must be** included in the substring.

If the begin/end indexes are out of **ccs** bounds, then function does nothing and returns NULL. Otherwise, the function allocates **JUST ENOUGH** space for the substring, copies the substring into the allocated space, and returns back the substring.

You have to print out the substring as well as the **ccs** after calling this function. After print-out don't forget to free allocated space for the returned substring.

Afterwards, go to step (1) and print the menu again.

(6) Else if the operation selection is (3), then ask for the beginning and end character indexes for removing the substring:

Enter the BEGIN INDEX and END INDEX numbers:

Implement the following function prototype for this operation:

char *remove_string(char **ccs, int begin_index, int end_index)

Note that **begin_index** and **end_index** are both INCLUSIVE and indexing starts from 0. That is, the characters at locations **begin_index** and **end_index must be** removed from the **ccs**.

If the begin/end indexes are out of **ccs** bounds, then function does nothing and returns NULL. Otherwise, the function allocates **JUST ENOUGH** space for the removed string, move the characters into allocated space, reallocates the **ccs** by the new string length, and returns back the removed **ccs** substring.

You have to print out the removed-string as well as the **ccs** after calling this function. After print-out don't forget to free allocated space for the returned remove string.

Afterwards, go to step (1) and print the menu again.

(7) Else if the operation selection is (4), then ask for the string to be inserted and the insert location:

Enter a new string for insertion:

Enter the BEGIN INDEX number where the new string begins:

Implement the following function prototype for this operation:

```
int insert_string(char **ccs, char *insert, int begin_index)
```

The **insert** will be inserted to the **ccs** starting from the **begin_index**. The length of the **ccs** will increase with the length of the **insert**. Again, memory must be reallocated so that new **ccs** exactly fits into the re-allocated space. Don't forget to free allocated space for the **insert** at the end of the insert_string() function.

The function returns the number of bytes that is allocated for new **ccs**. If the allocation is unsuccessful, then it returns -1.

You have to print out the **ccs** after calling this function.

Afterwards, go to step (1) and print the menu again.

(8) Else if the operation selection is (5), then ask for the string to be replaced and the string that will be used in replacement:

Find what:

Replace with:

Implement the following function prototype for this operation:

```
int replace_string(char **ccs, char *find, char *replace)
```

All the occurrences of the **find** in the ccs will be found and replaced with **replace**.

The length of the **ccs** can increase or decrease based on the lengths of **find** and **replace**. Again, memory must be reallocated so that new **ccs** exactly fits into the re-allocated space. Don't forget to free allocated space for the **find** and **replace** at the end of the replace_string() function.

The function returns number of replacements.

You have to print out how many replacement made and the **ccs** after calling this function.

Afterwards, go to step (1) and print the menu again.

Example Program Run:

0: Exit The Program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace

Enter Your Choice: 2

You have to set Current Character Sequence (CCS) before doing any operations!

0: Exit The Program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace

Enter Your Choice: 1
Enter CCS: A man, a plan, a canal, Panama!

0: Exit The Program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace

Enter Your Choice: 2
Enter the BEGIN INDEX and END INDEX numbers: 2 6

CCS: “A man, a plan, a canal, Panama!”
Substring(2,6): “man, “

0: Exit The Program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace

Enter Your Choice: 3
Enter the BEGIN INDEX and END INDEX numbers: 2 8

CCS: “A plan, a canal, Panama!”
Removed String(2,8): “man, a “

0: Exit The Program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace

Enter Your Choice: 4
Enter a new string for insertion: man, a

Enter the BEGIN INDEX number where the new string begins: 2
CCS: “A man, a plan, a canal, Panama!”

0: Exit The Program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace

Enter Your Choice: 5
Find what: A
Replace with: One

There were 1 replacement.
CCS: “One man, a plan, a canal, Panama!”

0: Exit The Program
1: Set Current Character Sequence
2: Get Substring
3: Remove Substring
4: Insert
5: Replace

Enter Your Choice: 0

Goodbye!