CSC3320 System Level Programming
Lab Assignment 10 - Post-Lab

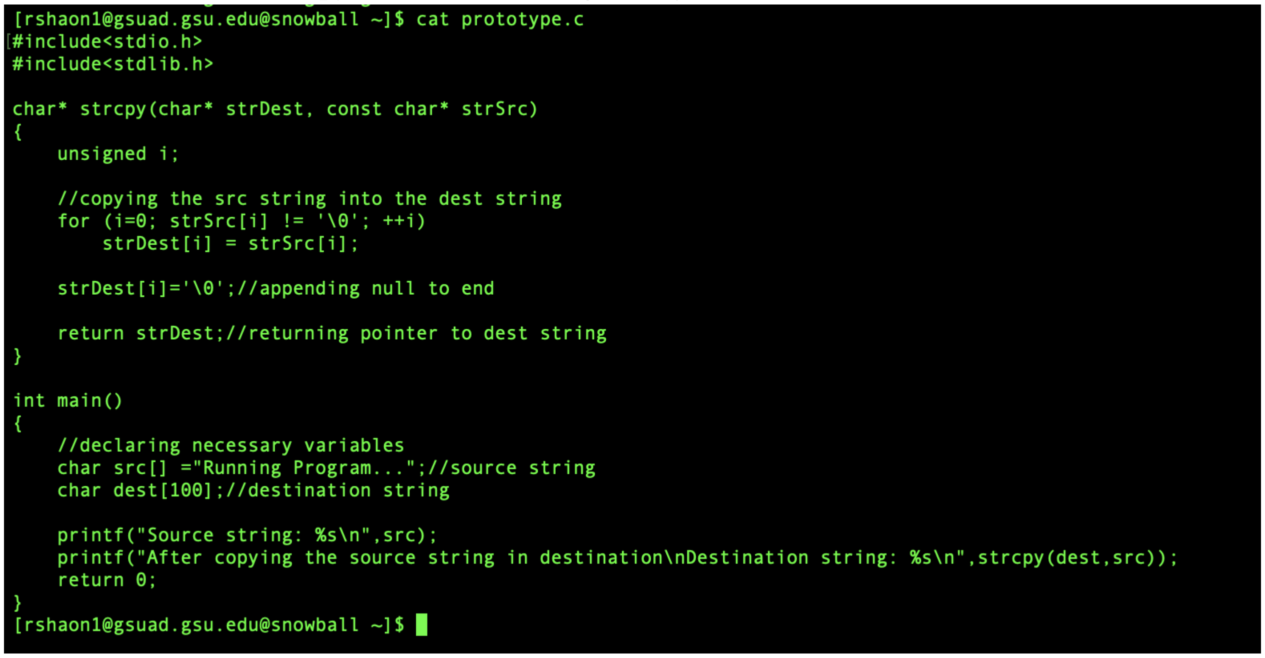
Rafid Shaon

Due at 11:59 pm on Friday, April 02, 2021

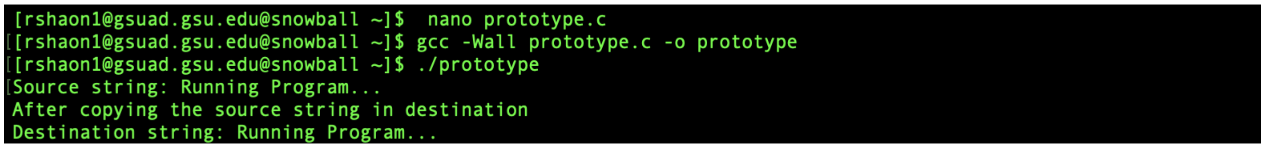
Purpose: Learn how to use the pointers to represent strings in C.

Part 1:
Write a function about string copy, the *strcpy* prototype "*char* strcpy (char* strDest, const char* strSrc*);". Here *strDest* is the destination string, *strSrc* is the source string.

1) Write the function *strcpy*, don't call C string library.



```
[rshaon1@gsuad.gsu.edu@snowball ~]$ cat prototype.c
#include<stdio.h>
#include<stdlib.h>

char* strcpy(char* strDest, const char* strSrc)
{
    unsigned i;

    //copying the src string into the dest string
    for (i=0; strSrc[i] != '\0'; ++i)
        strDest[i] = strSrc[i];

    strDest[i]='\0';//appending null to end

    return strDest;//returning pointer to dest string
}

int main()
{
    //declaring necessary variables
    char src[] ="Running Program...";//source string
    char dest[100];//destination string

    printf("Source string: %s\n",src);
    printf("After copying the source string in destination\nDestination string: %s\n",strcpy(dest,src));
    return 0;
}
[rshaon1@gsuad.gsu.edu@snowball ~]$
```



```
[rshaon1@gsuad.gsu.edu@snowball ~]$  nano prototype.c
[rshaon1@gsuad.gsu.edu@snowball ~]$ gcc -Wall prototype.c -o prototype
[rshaon1@gsuad.gsu.edu@snowball ~]$ ./prototype
Source string: Running Program...
After copying the source string in destination
Destination string: Running Program...
```

2) Here *strcpy* can copy *strSrc* to *strDest*, but why we use *char** as the return value of *strcpy*?

We use char* as a return type for two reasons: first, this prototype makes coding easier for nesting purposes like we can just call this function within the printf() function and the destination string would be printed. Another reason is that if any function returns some value than we can check whether this function works correctly or not by checking the return value; that means if(strcpy()==null) then within the function there must be something wrong, so like this error check can be done with the help of a return value.

Part 2:

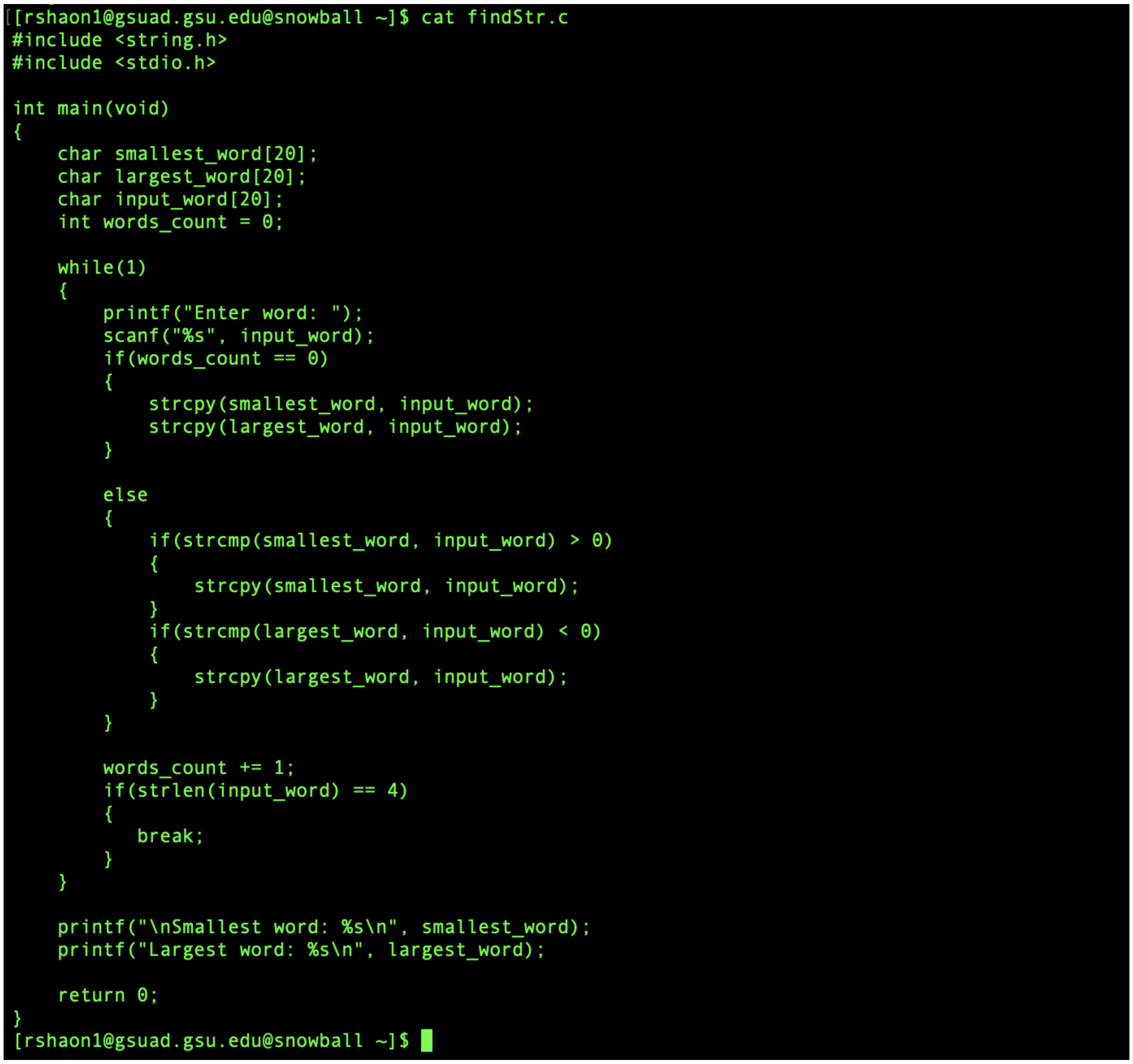
Write a program ***findStr.c*** that finds the "smallest" and "largest" in a series of words. After the user enters the words, the program will determine which words would come first and last if the words were listed in dictionary order. The program must stop accepting input when the user enters a four-letter word. Assume that no word is more than 20 letters long. An interactive session with the program might look like this:

```
Enter word: dog
Enter word: zebra
Enter word: rabbit
Enter word: catfish
Enter word: walrus
Enter word: cat
Enter word: fish

Smallest word: cat
Largest word: zebra
```

Hint: Use two strings named *smallest_word* and *largest_word* to keep track of the "smallest" and "largest" words entered so far. Each time the user enters a new word, use *strcmp* to compare it with the *smallest_word*; if the new word is "smaller", use *strcpy* to save it in the *smallest_word*. Do a similar comparison with *largest_word*. Use *strlen* to determine when the user has entered a four-letter word.

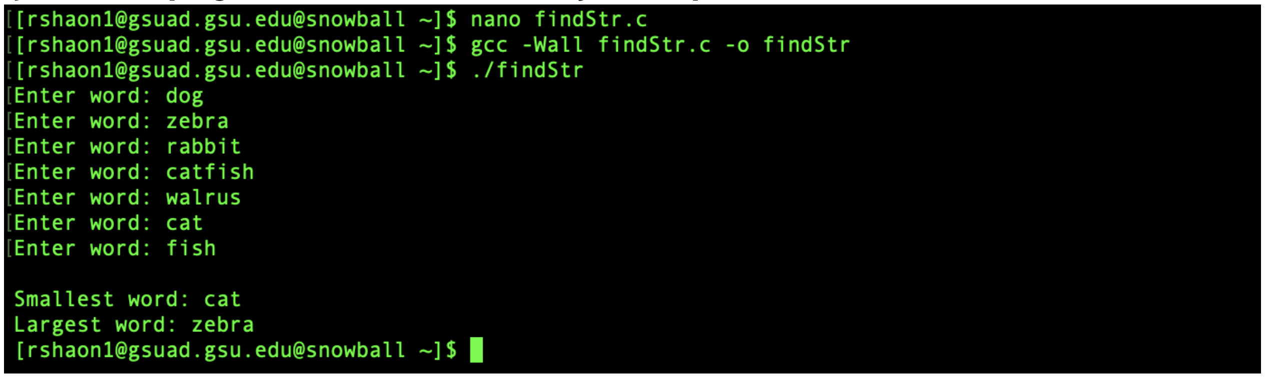
Questions:

1) Attach the source code of your C program into the answer sheet.



```
[[rshaon1@gsuad.gsu.edu@snowball ~]$ cat findStr.c
#include <string.h>
#include <stdio.h>

int main(void)
{
    char smallest_word[20];
    char largest_word[20];
    char input_word[20];
    int words_count = 0;

    while(1)
    {
        printf("Enter word: ");
        scanf("%s", input_word);
        if(words_count == 0)
        {
            strcpy(smallest_word, input_word);
            strcpy(largest_word, input_word);
        }

        else
        {
            if(strcmp(smallest_word, input_word) > 0)
            {
                strcpy(smallest_word, input_word);
            }
            if(strcmp(largest_word, input_word) < 0)
            {
                strcpy(largest_word, input_word);
            }
        }

        words_count += 1;
        if(strlen(input_word) == 4)
        {
           break;
        }
    }

    printf("\nSmallest word: %s\n", smallest_word);
    printf("Largest word: %s\n", largest_word);

    return 0;
}
[rshaon1@gsuad.gsu.edu@snowball ~]$
```

2) Run the C program, attach a screenshot of the output in the answer sheet.



```
[[rshaon1@gsuad.gsu.edu@snowball ~]$ nano findStr.c
[[rshaon1@gsuad.gsu.edu@snowball ~]$ gcc -Wall findStr.c -o findStr
[[rshaon1@gsuad.gsu.edu@snowball ~]$ ./findStr
[Enter word: dog
[Enter word: zebra
[Enter word: rabbit
[Enter word: catfish
[Enter word: walrus
[Enter word: cat
[Enter word: fish

Smallest word: cat
Largest word: zebra
[rshaon1@gsuad.gsu.edu@snowball ~]$
```

Submission:

- Please follow the instructions below step by step, and then write a report by answering the questions and upload the report (named as Lab10_FirstNameLastName.pdf or Lab10_FirstNameLastName.doc) to Google Classroom, under the rubric Lab 10 – Post Lab Assignment.
- Upload the C file findStr.c to the folder named "Lab 10 – Post Lab" in Google Classroom.
- Please add the lab assignment NUMBER and your NAME at the top of your filesheet.