Jamie Lopez

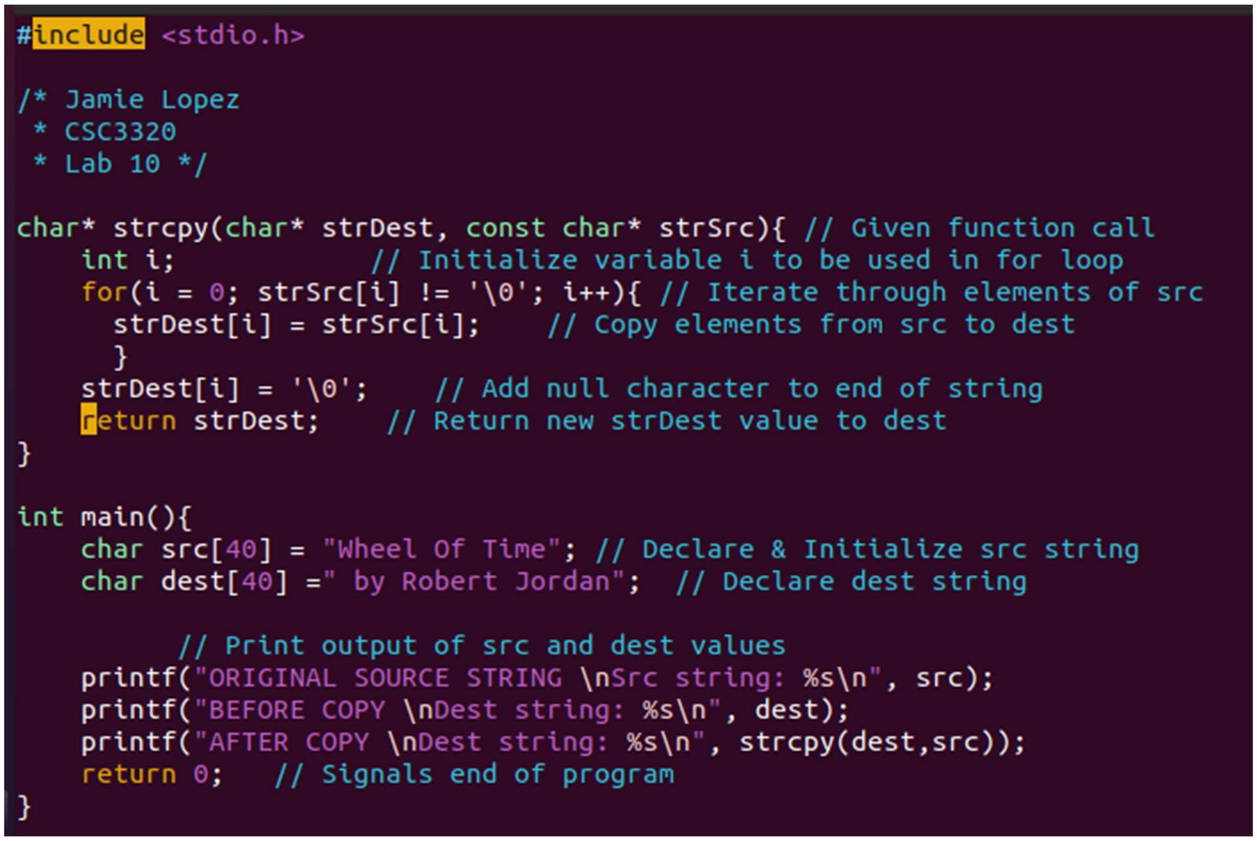
CSC3320

Lab 10 – Out Lab

Due 4/2/21

Part 1

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



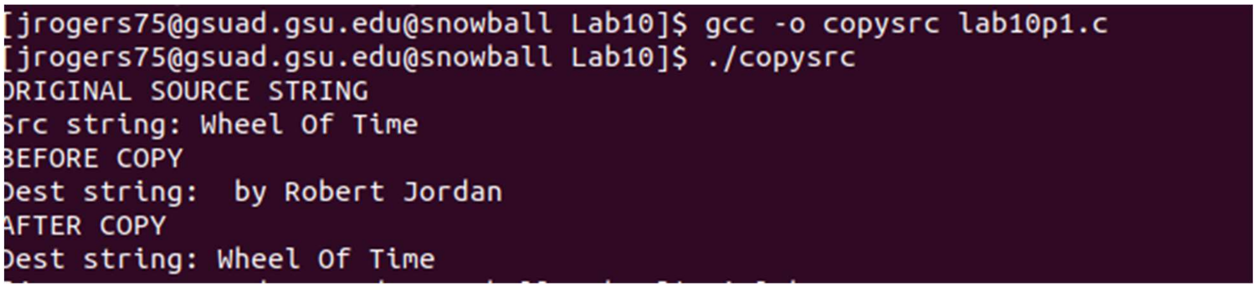
```
#include <stdio.h>

/* Jamie Lopez
 * CSC3320
 * Lab 10 */

char* strcpy(char* strDest, const char* strSrc){ // Given function call
    int i;              // Initialize variable i to be used in for loop
    for(i = 0; strSrc[i] != '\0'; i++){ // Iterate through elements of src
      strDest[i] = strSrc[i];    // Copy elements from src to dest
      }
    strDest[i] = '\0';    // Add null character to end of string
    return strDest;    // Return new strDest value to dest
}

int main(){
    char src[40] = "Wheel Of Time"; // Declare & Initialize src string
    char dest[40] =" by Robert Jordan";   // Declare dest string

          // Print output of src and dest values
    printf("ORIGINAL SOURCE STRING \nSrc string: %s\n", src);
    printf("BEFORE COPY \nDest string: %s\n", dest);
    printf("AFTER COPY \nDest string: %s\n", strcpy(dest,src));
    return 0;    // Signals end of program
}
```



```
[jrogers75@gsuad.gsu.edu@snowball Lab10]$ gcc -o copysrc lab10p1.c
[jrogers75@gsuad.gsu.edu@snowball Lab10]$ ./copysrc
ORIGINAL SOURCE STRING
Src string: Wheel Of Time
BEFORE COPY
Dest string:  by Robert Jordan
AFTER COPY
Dest string: Wheel Of Time
```

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

We use char* as the return value because it returns the location of the answer rather than returning its value. It makes it easier when nesting the function call. Also, it is points to the first character of the string and not the whole string. The pointer is the device used to send the whole string through rather than just the first character.

Part 2

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

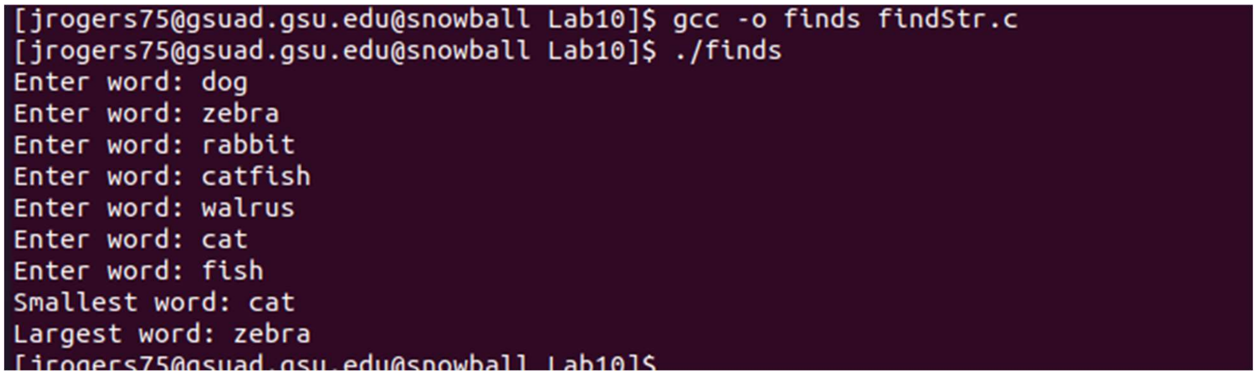
```
#include <stdio.h>
#include <string.h>
/* Jamie Lopez
 * CSC3320
 * Lab 10 - Part 2 */

int main()
{
    char smallest_word[20] = "z"; // Declare/Initialize smallest word variable
    char largest_word[20] = "";  // Declare/Initialize largest word variable
    char hold_word[20];         // Declare hold variable to compare with
    int status;                // Need a true condition to run while loop

    while(status = 1){        // Start while loop with true condition
    printf("Enter word: ");  // Prompt user to enter word
    scanf("%s", hold_word);  // Store user input in hold_word

   if(strcmp(smallest_word, hold_word) > 0){ //Compare hold to smallest
          strcpy(smallest_word, hold_word);  // If hold is smaller, copy it
          } else {                          // into smallest_word. Then,
     if(strcmp(largest_word, hold_word) < 0){// Compare hold to largest
              strcpy(largest_word, hold_word);  // If hold is larger, copy it
              }                                 // into largest_word
          }
       if(strlen(hold_word) == 4){     // Check length of word, if it's =4,
            break;                    // end the program
         }
     }
    printf("Smallest word: %s\n", smallest_word);//Print output for smallest
    printf("Largest word: %s\n", largest_word); //Print output for largest
    return 0;                // End of program
}
```

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



```
[jrogers75@gsuad.gsu.edu@snowball Lab10]$ gcc -o finds findStr.c
[jrogers75@gsuad.gsu.edu@snowball Lab10]$ ./finds
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
[jrogers75@gsuad.gsu.edu@snowball Lab10]$
```