[CS200]-Software Tools and Technologies II **Fall 2020-21**

Homework: IV

Team Name: Peaky Blinders Roll Numbers:11940150, 11940370, 11940380

Solution of problem 3. The program given in the question has debugged using gdb on Oracle Linux virtual machine on my PC. The gdb commands along with the explanation for the debugging process have been given below. Also the gdb log file is included in the submission.

```
Bug 1
gcc -g q3.c
gdb a.out
set logging on (Dump the log in gdb.txt)
run (segmentation fault observed)
backtrace (segmentation fault backtraced into Init_CString function)
(Error occurs when copying str to p->str)
break q3.c:14
(breakpoint added at line 14 in code where sigsev occurs i.e. strncpy function)
run (run program again from beginning till breakpoint)
print str
print p->len (see if these variables have proper value)
(No problem with above variables)
print p->str (<error: Cannot access memory at address 0xbaadf00d> message received)
(The above error occurs because the dynamically created p->str pointer is not
pointing to a valid string)
(Correction: Make p->str point to a string by allocating it one string of p->len
using malloc)
quit

Bug 2
gcc -g q3.c
gdb a.out
(Recompile and and start debugging again to check if bug is resolved and find
other bugs.)
set logging on
run
(still sigsev this time but at the end after ouput is printed and the output
seems wrong.)
(The output for Init seems okay but for chomp and append it consists of some
random characters.)
(Hence let us check Chomp function first.)
break q3.c:29
break q3.c:31 (Add breakpoint to check the contents of the variables.)
run (run program again till breakpoint)
print lastchar
print cstring->str
```

```
continue (till next breakpoint)
print cstring->str
(It is seen that instead of removing the last character of string i.e. '!' we are
actually adding '0' to the string. Also the lastchar must contain '!' but contains
'\0' instead.)
(Correction: Adjust the pointer so that it points to '!' and not '\0' by moving it
one position back. Also now with pointer pointing to right position we change its
content from '!' to '\0' and not '0')

Bug 3
continue
(After both break points. We see that append still has wrong output and sigsev at
the end)
clear
break q3.c:42
break q3.c:44 (Check value of variables in Append function)
run (run program again from beginning till breakpoint)
print p->len
print newstr
continue
print p->str
(Wrong length string is allocated to newstr. Also empty character has to be
taken care of.)
(Correction: Allocate a string to newstr after updating p->len. The '\0'
automatically appended by snprintf, we just have to increase the length passed by
1 to accomodate for '\0'.)
quit

Bug 4
gcc -g q3.c
gdb a.out
set logging on
(Recompile and and start debugging again to check if bug is resolved and find
other bugs.)
run (This time output is correct but sigsev at the end.)
where (It appears to be coming from Delete_CString function)
(The problem is that we free p and then we free p->str. But due to first free p
is not pointing to anything and hence trying to access the contents of the location
it points to gives error.)
(Correction: free(p->str) should be done before free(p).)
quit

Final check
gcc -g q3.c
gdb a.out
set logging on
(Recompile and and start debugging again to check if bug is resolved and find
other bugs.)
run (This time gdb does not find any errors.)
quit
(Once running the program normally. It executes and terminates properly.)
```

```
(Hence all the bugs have been fixed using gdb.)
```

After debugging the final code is:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct
{
    char *str;
    int len;
} CString;

CString *Init_CString(char *str)
{
    CString *p = malloc(sizeof(CString));
    p->len = strlen(str);
    p->str = malloc(p->len + 1); //Bug 1 corrected
    strncpy(p->str, str, strlen(str) + 1);
    return p;
}
void Delete_CString(CString *p)
{
    free(p->str); //Bug 4 corrected
    free(p);
}
// Removes the last character of a CString and returns it.
//
char Chomp(CString *cstring)
{
    char lastchar = *(cstring->str + cstring->len - 1); //Bug 2 corrected
    // Shorten the string by one
    *(cstring->str + cstring->len - 1) = '\0'; //Bug 2 corrected
    cstring->len = strlen(cstring->str);
    return lastchar;
}
// Appends a char * to a CString
//
CString *Append_Chars_To_CString(CString *p, char *str)
{
    p->len = p->len + strlen(str);
    char *newstr = malloc(p->len + 1); //Bug 3 corrected
    // Create the new string to replace p->str
    snprintf(newstr, p->len + 1, "%s%s", p->str, str); //Bug 3 corrected
    // Free old string and make CString point to the new string
    free(p->str);
    p->str = newstr;
    return p;
}
```

```
int main(void)
{
    CString *mystr;
    char c;
    mystr = Init_CString("Hello!");
    printf("Init:\n str: `%s' len: %d\n", mystr->str, mystr->len);
    c = Chomp(mystr);
    printf("Chomp '%c':\n str:`%s' len: %d\n", c, mystr->str, mystr->len);
    mystr = Append_Chars_To_CString(mystr, " world!");
    printf("Append:\n str: `%s' len: %d\n", mystr->str, mystr->len);
    Delete_CString(mystr);
    return 0;
}
```

Comments of the form *Bug x corrected* where is x is the bug number given in chronological order of correction are given where the fix has been done.