#include <stdio.h>

#include <stdlib.h>

#include <wchar.h>

#define PASSWORD "ABCD1234!"

/\*You need not worry about other include statements if at all any are missing \*/

void func1()

{

char \* data;

char \* dataBuffer = (char \*)ALLOCA(100\*sizeof(char));

memset(dataBuffer, 'A', 100-1);

dataBuffer[100-1] = '\0';

data = dataBuffer - 8;

{

char source[100];

memset(source, 'C', 100-1);

source[100-1] = '\0';

strcpy(data, source);

if(data != NULL)

{

printf("%s\n", data);

}

}

}

* Function 1 prints C from 0-99 times.
* Data has the value of ‘C’ as all the elements.
* The scope of C is from line 15-24 only.
* So whatever happens out of it, doesn’t matter because printf() is inside this scope.

void func2()

{

char \* data;

data = NULL;

data = (char \*)calloc(100, sizeof(char));

strcpy(data, "A String");

if(data != NULL)

{

printf("%s\n", data);

}

}

* Function 2 create a data with char as 100 blocks.
* It is initialize to zero.
* “A string” is copied to data.
* Thus it prints “A String”.

void func3()

{

char \* password;

char passwordBuffer[100] = "";

password = passwordBuffer;

strcpy(password, PASSWORD);

{

HANDLE pHandle;

char \* username = "User";

char \* domain = "Domain";

/\* Let's say LogonUserA is a custon authentication function\*/

if (LogonUserA(

username,

domain,

password,

&pHandle) != 0)

{

printf("User logged in successfully.\n");

CloseHandle(pHandle);

}

else

{

printf("Unable to login.\n");

}

}

}

* Function 3 seems to be fine.
* Initially I thought, why we need line 58 and 59. Why can’t we do 57 and 60.
* But I did some trails and found that we cannot copy.
* Strcpy works on 2 memory blocks whereas, argv 1 is pointer.
* Thus we are assigning memory block which is pointer by that pointer.

static void func4()

{

char \* data;

data = NULL;

data = (char \*)calloc(20, sizeof(char));

if (data != NULL)

{

strcpy(data, "Initialize");

if(data != NULL)

{

printf("%s\n", data);

}

free(data);

}

}

* My understanding:
  + Function 4 will print “Initialize”.
  + Calloc will initialize a zero buffer.
  + Thus, true to both the if conditions.
* As per hands-on:
  + First if condition of line 95 is not true. Thus, no output. I’m not sure why!!! Need to check..

void func5()

{

int i = 0;

do

{

printf("%d\n", i);

i = (i + 1) % 256;

} while(i >= 0);

}

* Function 5, it will go to infinite loop.
* It starts with checking 255, 254, 253, …, 0, 255, 254, 253, …, 0
* “=0” condition makes it infinite.

void func6()

{

char dataBuffer[100] = "";

char \* data = dataBuffer;

printf("Please enter a string: ");

if (fgets(data, 100, stdin) < 0)

{

printf("fgets failed!\n");

exit(1);

}

if(data != NULL)

{

printf("%s\n", data);

}

}

* Function 6 seems to be fine.

void func7()

{

char \* data;

data = "Fortify";

data = NULL;

printf("%s\n", data);

}

* Function 7 will give some error. Because we are changing the address tore in pointer to NULL.

This main function calls 7 functions internally.

int main(int argc, char \* argv[])

{

printf("Calling func1\n");

func1();

printf("Calling func2\n");

func2();

printf("Calling func3\n");

func3();

printf("Calling func4\n");

func4();

printf("Calling func5\n");

func5();

printf("Calling func6\n");

func6();

printf("Calling func7\n");

func7();

return 0;

}