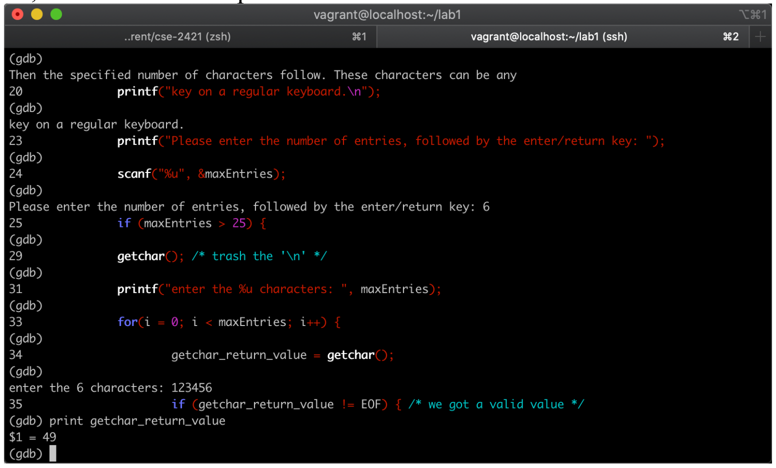
THIS IS THE README FILE FOR LAB 1.

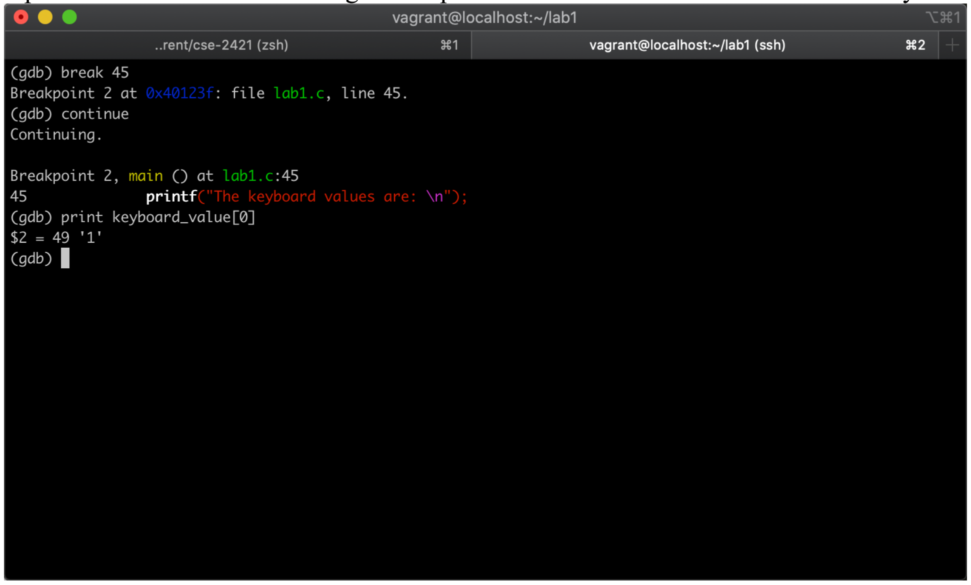
Name: Yifan Yao.740

1. UNIX is proprietary system (i.e. you must purchase a license) while Linux is an Open Source system. An Open Source system, however, is not always "free". Why?
    a. Although Linux is free to use for everyone, but in production environment, it is not free to get support from distributer (for immediate access to crucial security patch and reliability problems).
2. Name another difference between Unix/Linux from your required reading.
    a. Linux and Unix are using different file system, and everyone can build their own kernel as they wish.
3. What made UNIX different from all other commercially available software systems when it was initially developed?
    a. Other commercially available software was specifically designed for one system, but Unix kernel is able to adapt to every specific system.
4. What operating system covers the widest range of hardware in the world?
    a. Linux
5. Why was the C programming language initially created?
    a. To "create the UNIX system".
6. How is white space (tabs/newlines/returns, spaces, etc.) handled within a C program file?
    a. "Ignored by compiler for the most part."
7. What did you learn when you "mistyped" (e.g. "fat fingered") the compile command?
    a. Create multiple copy of your source code or use version-control system.
8. When a line of code is printed out after the **next** command in gdb, that line of code has just finished executing. True or False?
    a. False
9. What is the value of **maxEntries** variable at GDB item #3?
    a. 6
10. What is the value of **getchar_return_value** at GDB item #4? Is this what you expected to see?
    a. 49, Yes because 49 represents '1' in ASCII.
    b. 

```
(gdb)
Then the specified number of characters follow. These characters can be any
20              printf("key on a regular keyboard.\n");
(gdb)
key on a regular keyboard.
23              printf("Please enter the number of entries, followed by the enter/return key: ");
(gdb)
24              scanf("%u", &maxEntries);
(gdb)
Please enter the number of entries, followed by the enter/return key: 6
25              if (maxEntries > 25) {
(gdb)
29              getchar(); /* trash the '\n' */
(gdb)
31              printf("enter the %u characters: ", maxEntries);
(gdb)
33              for(i = 0; i < maxEntries; i++) {
(gdb)
34                      getchar_return_value = getchar();
(gdb)
enter the 6 characters: 123456
35                      if (getchar_return_value != EOF) { /* we got a valid value */
(gdb) print getchar_return_value
$1 = 49
(gdb) █
```

11. What is the value of **keyboard_value[0]** at GDB item #5? Did gdb give you additional information this time? What did you find out when you referred to the Extended ASCII table? Did you have an "Aha!" moment? What was it?
    a. 49 '1', 1 is corresponding with 49 in ASCII table, Yes gdb gives me additional information (it converted the ASCII to character for me), the getchar_return_value will only return the character's

representation in ASCII but the gdb will provide conversion for value storied in array.

b. 

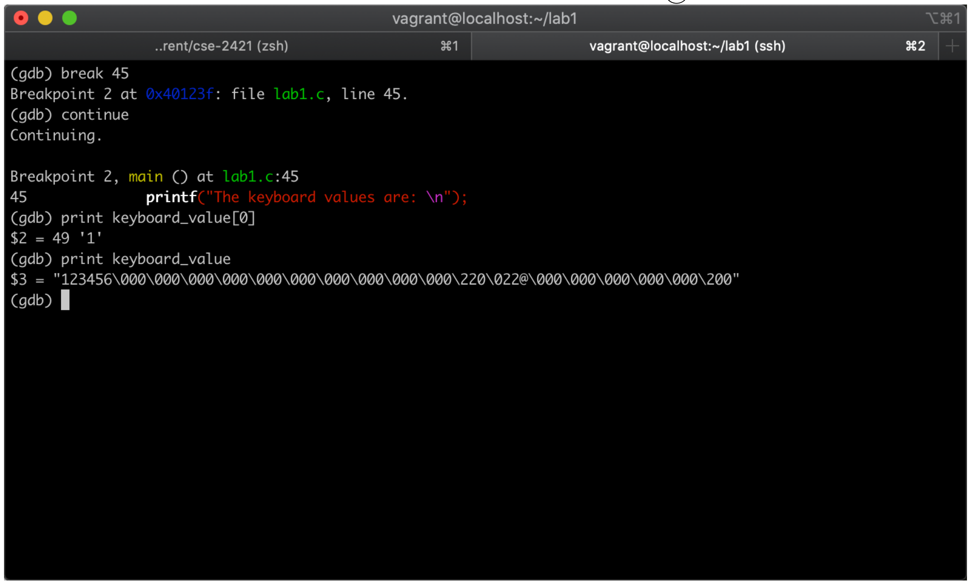
```
(gdb) break 45
Breakpoint 2 at 0x40123f: file lab1.c, line 45.
(gdb) continue
Continuing.

Breakpoint 2, main () at lab1.c:45
45                printf("The keyboard values are: \n");
(gdb) print keyboard_value[0]
$2 = 49 '1'
(gdb)
```

12. What are the values in the first 6 keyboard_value elements?

keyboard_value[0]= 1
keyboard_value[1]= 2
keyboard_value[2]= 3
keyboard_value[3]= 4
keyboard_value[4]= 5
keyboard_value[5]= 6

13. How is the output different when the **print keyboard_value** instruction is used?

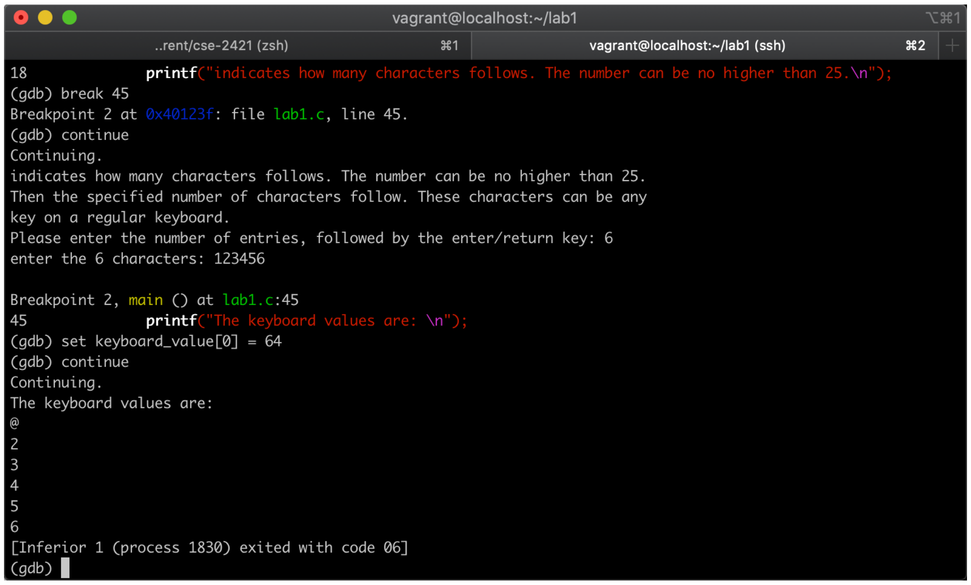
a. "123456\000\000\000\000\000\000\000\000\000\000\220\022@\000\000\000\000\000\200"

b. 

```
(gdb) break 45
Breakpoint 2 at 0x40123f: file lab1.c, line 45.
(gdb) continue
Continuing.

Breakpoint 2, main () at lab1.c:45
45                printf("The keyboard values are: \n");
(gdb) print keyboard_value[0]
$2 = 49 '1'
(gdb) print keyboard_value
$3 = "123456\000\000\000\000\000\000\000\000\000\000\220\022@\000\000\000\000\000\200"
(gdb)
```

14. How does the output of the program change at GDB item #6 from the output seen prior?

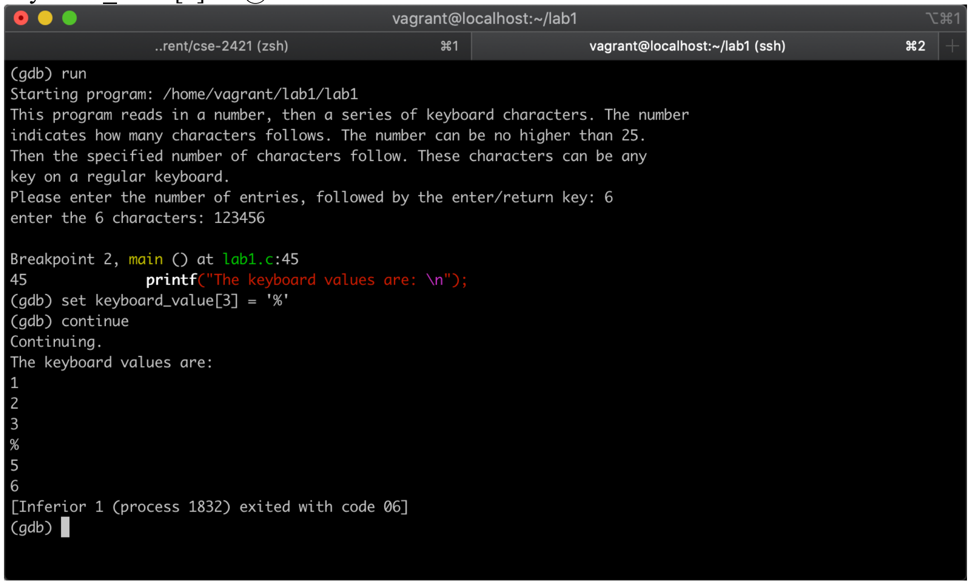
a. The first character should be ‘1’, but it has been changed to ‘@’ by setting “set keyboard_value[0] = 64”



```
vagrant@localhost:~/lab1
..rent/cse-2421 (zsh)    ⌘1    vagrant@localhost:~/lab1 (ssh)    ⌘2
18              printf("indicates how many characters follows. The number can be no higher than 25.\n");
(gdb) break 45
Breakpoint 2 at 0x40123f: file lab1.c, line 45.
(gdb) continue
Continuing.
indicates how many characters follows. The number can be no higher than 25.
Then the specified number of characters follow. These characters can be any
key on a regular keyboard.
Please enter the number of entries, followed by the enter/return key: 6
enter the 6 characters: 123456

Breakpoint 2, main () at lab1.c:45
45              printf("The keyboard values are: \n");
(gdb) set keyboard_value[0] = 64
(gdb) continue
Continuing.
The keyboard values are:
@
2
3
4
5
6
[Inferior 1 (process 1830) exited with code 06]
(gdb)
```

b.

15. How does the output of the program change at GDB item #7 from the output seen prior?
    a. The fourth character should be '4', but it has been changed to '@' by setting "set keyboard_value[4] = '@'"



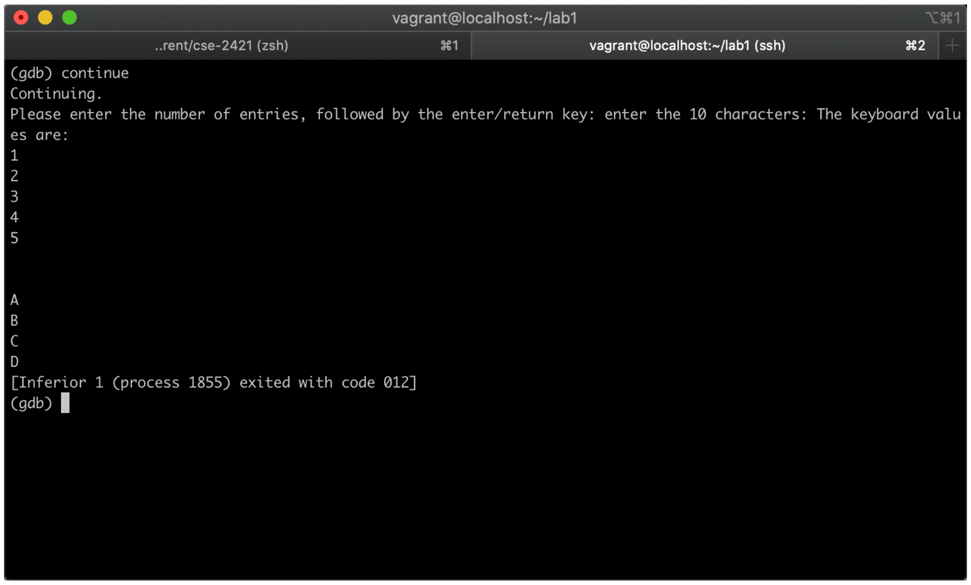
```
vagrant@localhost:~/lab1
..rent/cse-2421 (zsh)    ⌘1    vagrant@localhost:~/lab1 (ssh)    ⌘2
(gdb) run
Starting program: /home/vagrant/lab1/lab1
This program reads in a number, then a series of keyboard characters. The number
indicates how many characters follows. The number can be no higher than 25.
Then the specified number of characters follow. These characters can be any
key on a regular keyboard.
Please enter the number of entries, followed by the enter/return key: 6
enter the 6 characters: 123456

Breakpoint 2, main () at lab1.c:45
45              printf("The keyboard values are: \n");
(gdb) set keyboard_value[3] = '%'
(gdb) continue
Continuing.
The keyboard values are:
1
2
3
%
5
6
[Inferior 1 (process 1832) exited with code 06]
(gdb)
```

b.

16. How does the output of the program change at GDB item #9 from the output seen prior? What are the values in the first 11 **keyboard_value** elements?

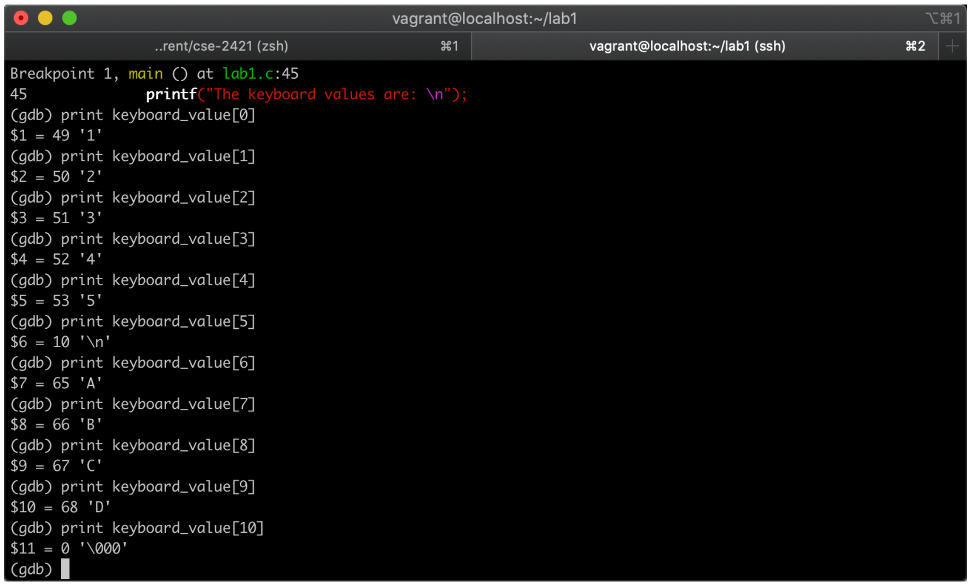
1
2
3
4
5

A
B
C
D



```
(gdb) continue
Continuing.
Please enter the number of entries, followed by the enter/return key: enter the 10 characters: The keyboard valu
es are:
1
2
3
4
5


A
B
C
D
[Inferior 1 (process 1855) exited with code 012]
(gdb) 
```

keyboard_value[0]= 49 '1'
keyboard_value[1]= 50 '2'
keyboard_value[2]= 51 '3'
keyboard_value[3]= 52 '4'
keyboard_value[4]= 53 '5'
keyboard_value[5]= 10 '\n'
keyboard_value[6]= 65 'A'
keyboard_value[7]= 66 'B'
keyboard_value[8]= 67 'C'
keyboard_value[9]= 68 'D'
keyboard_value[10]= 0 '\000'



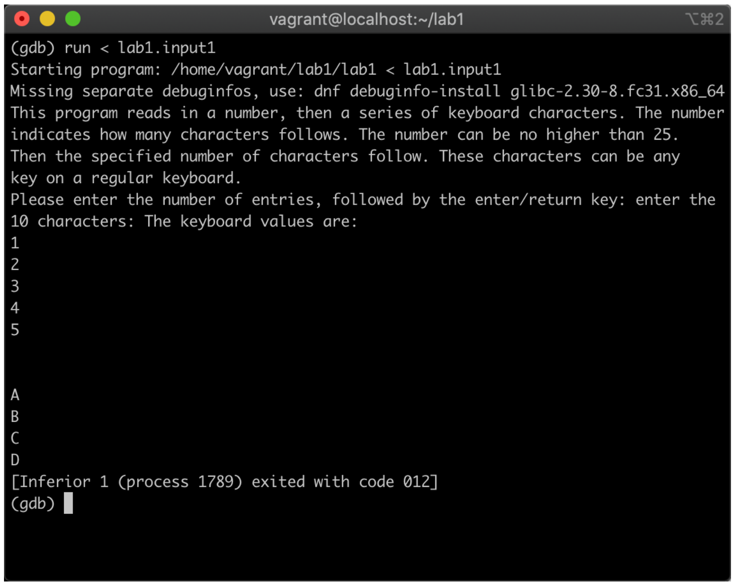
```
Breakpoint 1, main () at lab1.c:45
45              printf("The keyboard values are: \n");
(gdb) print keyboard_value[0]
$1 = 49 '1'
(gdb) print keyboard_value[1]
$2 = 50 '2'
(gdb) print keyboard_value[2]
$3 = 51 '3'
(gdb) print keyboard_value[3]
$4 = 52 '4'
(gdb) print keyboard_value[4]
$5 = 53 '5'
(gdb) print keyboard_value[5]
$6 = 10 '\n'
(gdb) print keyboard_value[6]
$7 = 65 'A'
(gdb) print keyboard_value[7]
$8 = 66 'B'
(gdb) print keyboard_value[8]
$9 = 67 'C'
(gdb) print keyboard_value[9]
$10 = 68 'D'
(gdb) print keyboard_value[10]
$11 = 0 '\000'
(gdb) 
```

Are these values what you expected to see? If so, why? If not, what did you observe/learn?

Yes, I was expected to see these values, because '\n' also count as a character.

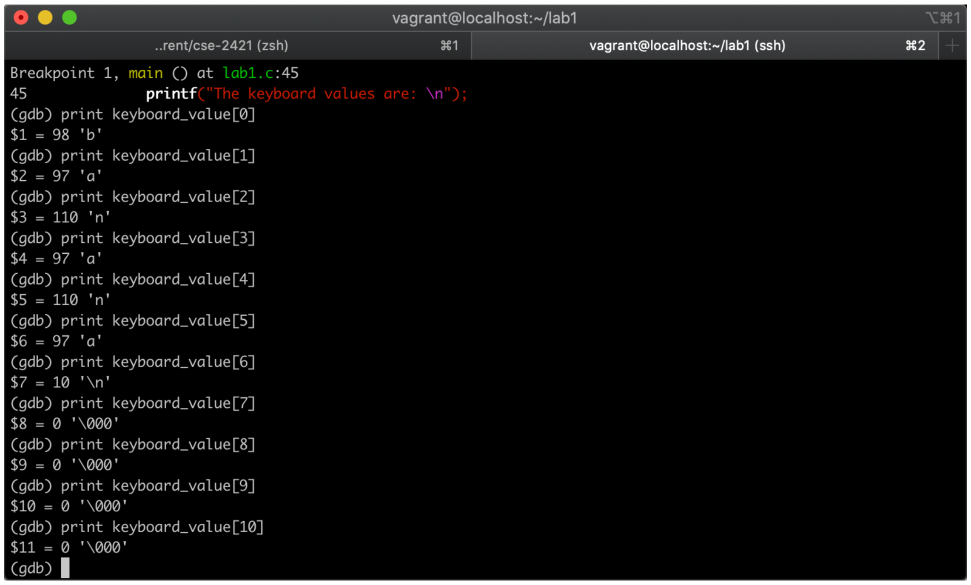
17. How does the output of the program change at GDB item #10 from the output seen prior? What are the values in the first 11 keyboard_value elements?

b
a
n
a
n
a



```
(gdb) run < lab1.input1
Starting program: /home/vagrant/lab1/lab1 < lab1.input1
Missing separate debuginfos, use: dnf debuginfo-install glibc-2.30-8.fc31.x86_64
This program reads in a number, then a series of keyboard characters. The number
indicates how many characters follows. The number can be no higher than 25.
Then the specified number of characters follow. These characters can be any
key on a regular keyboard.
Please enter the number of entries, followed by the enter/return key: enter the
10 characters: The keyboard values are:
1
2
3
4
5


A
B
C
D
[Inferior 1 (process 1789) exited with code 012]
(gdb) 
```

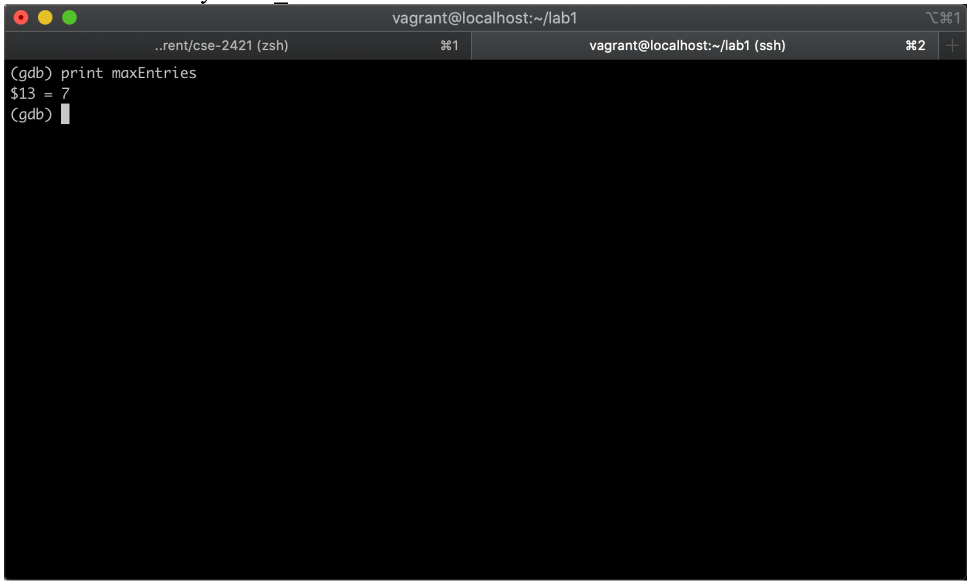
keyboard_value[0]= 98 'b'
keyboard_value[1]= 97 'a'
keyboard_value[2]= 110 'n'
keyboard_value[3]= 97 'a'
keyboard_value[4]= 110 'n'
keyboard_value[5]= 97 'a'
keyboard_value[6]= 10 '\n'
keyboard_value[7]= 0 '\000'
keyboard_value[8]= 0 '\000'
keyboard_value[9]= 0 '\000'
keyboard_value[10]= 0 '\000'



```
Breakpoint 1, main () at lab1.c:45
45              printf("The keyboard values are: \n");
(gdb) print keyboard_value[0]
$1 = 98 'b'
(gdb) print keyboard_value[1]
$2 = 97 'a'
(gdb) print keyboard_value[2]
$3 = 110 'n'
(gdb) print keyboard_value[3]
$4 = 97 'a'
(gdb) print keyboard_value[4]
$5 = 110 'n'
(gdb) print keyboard_value[5]
$6 = 97 'a'
(gdb) print keyboard_value[6]
$7 = 10 '\n'
(gdb) print keyboard_value[7]
$8 = 0 '\000'
(gdb) print keyboard_value[8]
$9 = 0 '\000'
(gdb) print keyboard_value[9]
$10 = 0 '\000'
(gdb) print keyboard_value[10]
$11 = 0 '\000'
(gdb)
```

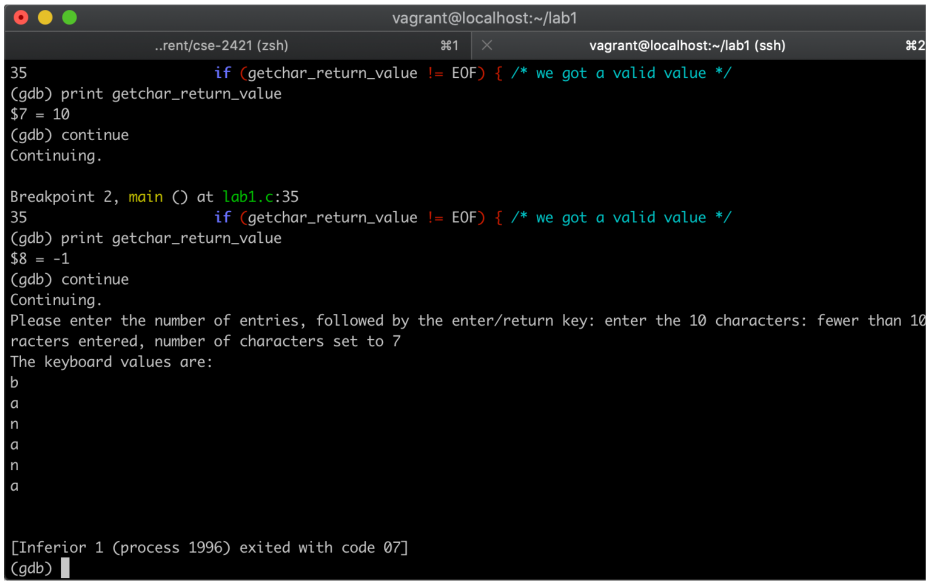
Are these values what you expected to see? If so, why? If not, what did you observe/learn?

a. Yes, I was expected to see these values, because there are only 7 character were inputted in this case, it is normal to leave array position 7 to 10 empty.

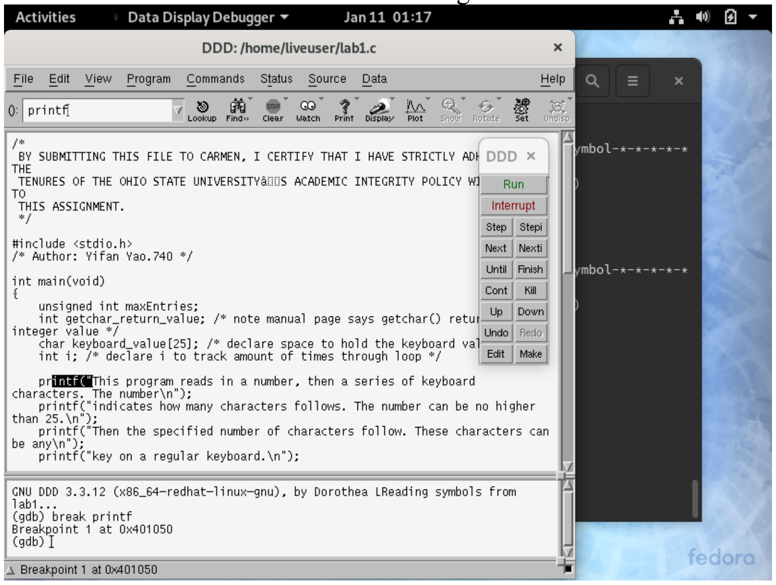
18. What is the value of **maxEntries** in GDB item #10? Why is this the correct value?
    a. 7, when there are no sufficient characters provided into the loop, the maxEntries will set to the total number of keyboard_value.
    b. 

```
(gdb) print maxEntries
$13 = 7
(gdb)
```

19. Run lab1 through gdb using lab1.input2 a second time, but set a breakpoint at "**if (getchar_return_value != EOF){**" (line 29 in my program) and check the value in **getchar_return_value** each time through the loop. What value does it have the last time through the loop? The value in the extended ASCII table says 255 represents "Latin small letter y with diaeresis" is that what we got? Might -1 mean something else in this case? Might the value not be a char? Use the linux command **man getchar** to help you answer this question.
    a. -1, the value in the extended ASCII table says 255 represents "non-breaking space", -1 might means not an ASCII character, from man-page shows that end of file (EOF) or error.

b. 

```
35                      if (getchar_return_value != EOF) { /* we got a valid value */
(gdb) print getchar_return_value
$7 = 10
(gdb) continue
Continuing.

Breakpoint 2, main () at lab1.c:35
35                      if (getchar_return_value != EOF) { /* we got a valid value */
(gdb) print getchar_return_value
$8 = -1
(gdb) continue
Continuing.
Please enter the number of entries, followed by the enter/return key: enter the 10 characters: fewer than 10
racters entered, number of characters set to 7
The keyboard values are:
b
a
n
a
n
a


[Inferior 1 (process 1996) exited with code 07]
(gdb)
```

20. Describe how you created the breakpoint in your program while using **ddd** instead of **gdb**.
    a. Select the statement and click the STOP sign from tool bar.
    b. 

```
/*
 BY SUBMITTING THIS FILE TO CARMEN, I CERTIFY THAT I HAVE STRICTLY ADH
THE
 TENURES OF THE OHIO STATE UNIVERSITYâS ACADEMIC INTEGRITY POLICY WI
TO
 THIS ASSIGNMENT.
 */

#include <stdio.h>
/* Author: Yifan Yao.740 */

int main(void)
{
    unsigned int maxEntries;
    int getchar_return_value; /* note manual page says getchar() retur
integer value */
    char keyboard_value[25]; /* declare space to hold the keyboard val
    int i; /* declare i to track amount of times through loop */

    printf("This program reads in a number, then a series of keyboard
characters. The number\n");
    printf("indicates how many characters follows. The number can be no higher
than 25.\n");
    printf("Then the specified number of characters follow. These characters can
be any\n");
    printf("key on a regular keyboard.\n");
```

```
GNU DDD 3.3.12 (x86_64-redhat-linux-gnu), by Dorothea LReading symbols from
lab1...
(gdb) break printf
Breakpoint 1 at 0x401050
(gdb)
```

```
Breakpoint 1 at 0x401050
```

21. Have you ever used the gdb/ddd debugger before? If so, for what class(es)?
    a. Yes, for Intro to C++.