CS256 ASSIGNMENT-4

UNIX AND SHELL SCRIPTING LAB

ROLL NO -422151

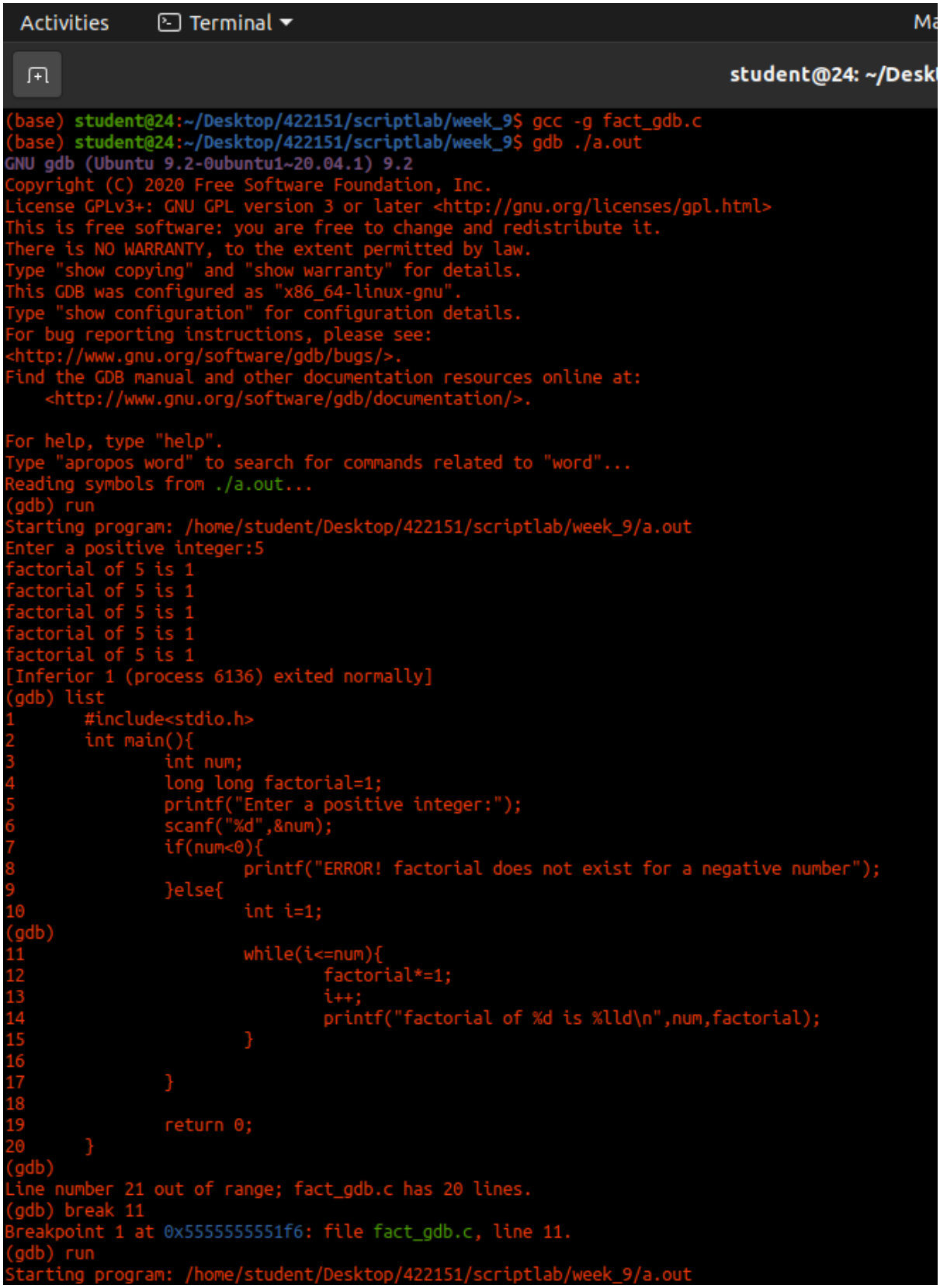
SEC-A

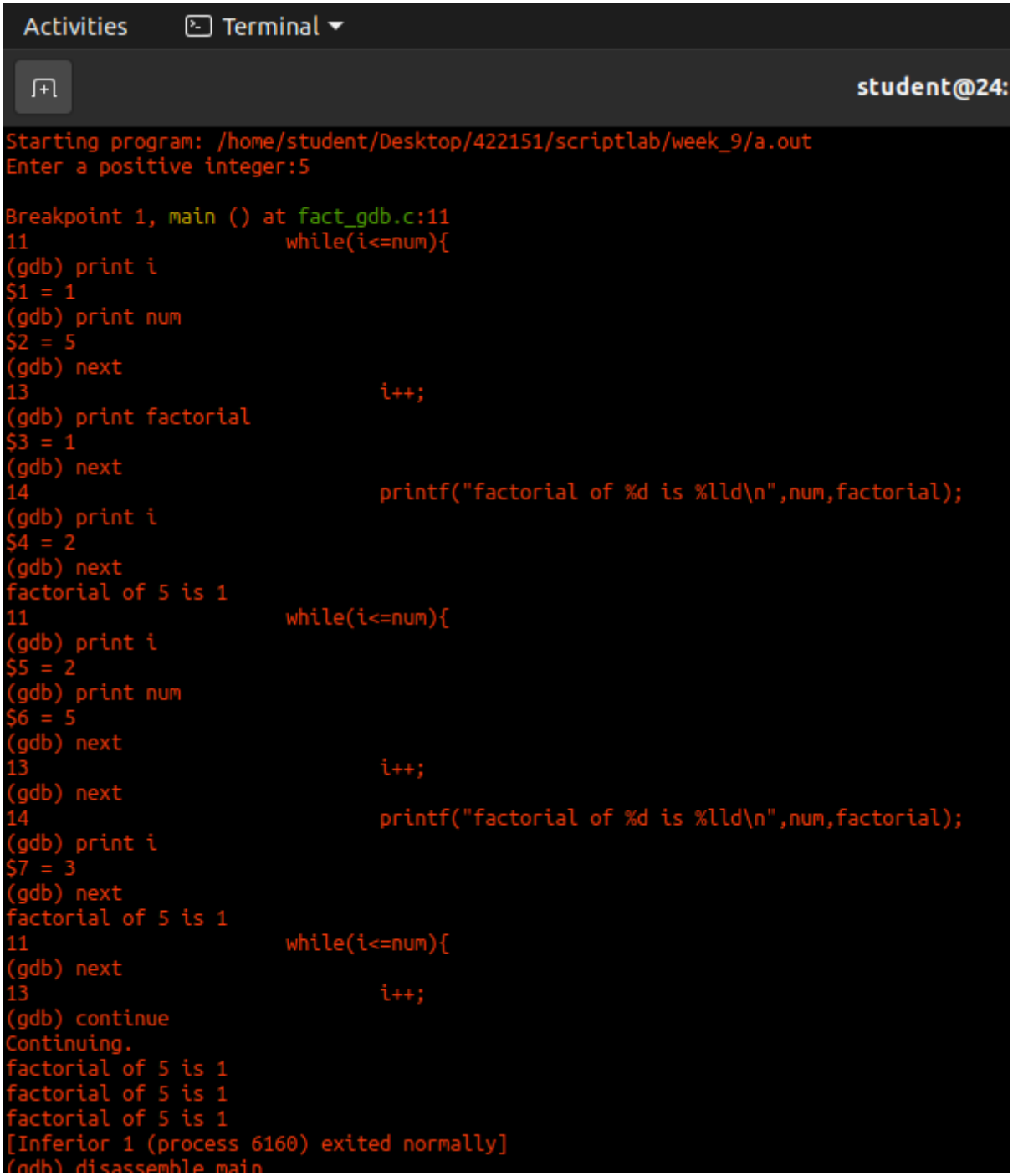
Generate different C programs that induce a segmentation fault error, select these examples of your choice, and employ the GDB utility for debugging on Linux.

fact_gdb.c

```c
#include<stdio.h>
int main(){
    int num;
    long long factorial=1;
    printf("Enter a positive integer:");
    scanf("%d",&num);
    if(num<0){
        printf("ERROR! factorial does not exist for a negative number");
    }else{
        int i=1;
        while(i<=num){
            factorial*=1;
            i++;
            printf("factorial of %d is %lld\n",num,factorial);
        }

    }

    return 0;
}
```

Output:


Activities
Terminal
student@24: ~/Desk
(base) student@24:~/Desktop/422151/scriptlab/week_9$ gcc -g fact_gdb.c
(base) student@24:~/Desktop/422151/scriptlab/week_9$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) run
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out
Enter a positive integer:5
factorial of 5 is 1
factorial of 5 is 1
factorial of 5 is 1
factorial of 5 is 1
factorial of 5 is 1
[Inferior 1 (process 6136) exited normally]
(gdb) list
1 #include<stdio.h>
2 int main(){
3 int num;
4 long long factorial=1;
5 printf("Enter a positive integer:");
6 scanf("%d",&num);
7 if(num<0){
8 printf("ERROR! factorial does not exist for a negative number");
9 }else{
10 int i=1;
(gdb)
11 while(i<=num){
12 factorial*=1;
13 i++;
14 printf("factorial of %d is %lld\n",num,factorial);
15 }
16
17 }
18
19 return 0;
20 }
(gdb)
Line number 21 out of range; fact_gdb.c has 20 lines.
(gdb) break 11
Breakpoint 1 at 0x5555555551f6: file fact_gdb.c, line 11.
(gdb) run
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out

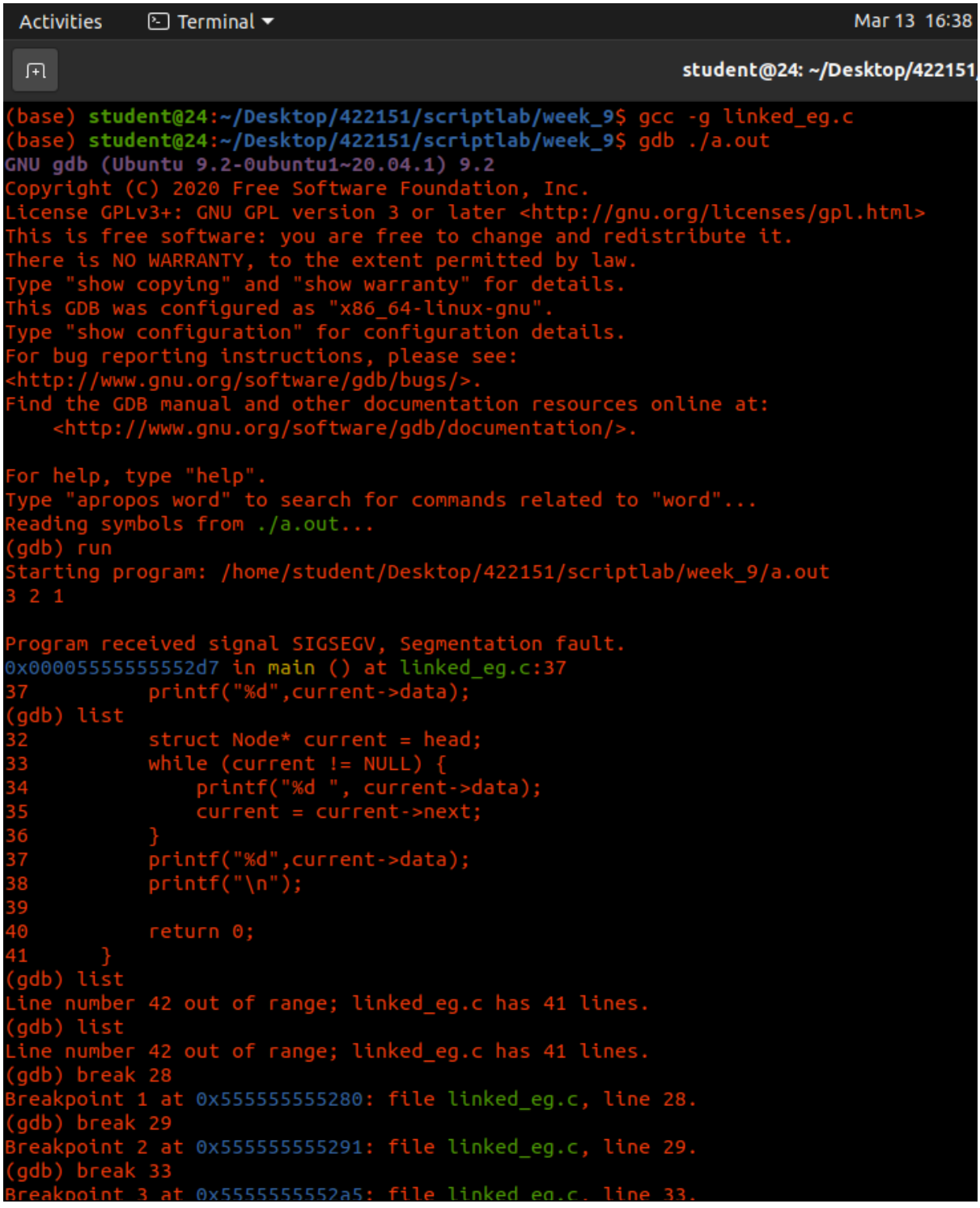

Activities
Terminal
student@24:
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out
Enter a positive integer:5
Breakpoint 1, main () at fact_gdb.c:11
11 while(i<=num){
(gdb) print i
$1 = 1
(gdb) print num
$2 = 5
(gdb) next
13 i++;
(gdb) print factorial
$3 = 1
(gdb) next
14 printf("factorial of %d is %lld\n",num,factorial);
(gdb) print i
$4 = 2
(gdb) next
factorial of 5 is 1
11 while(i<=num){
(gdb) print i
$5 = 2
(gdb) print num
$6 = 5
(gdb) next
13 i++;
(gdb) next
14 printf("factorial of %d is %lld\n",num,factorial);
(gdb) print i
$7 = 3
(gdb) next
factorial of 5 is 1
11 while(i<=num){
(gdb) next
13 i++;
(gdb) continue
Continuing.
factorial of 5 is 1
factorial of 5 is 1
factorial of 5 is 1
[Inferior 1 (process 6160) exited normally]
(gdb) disassemble main

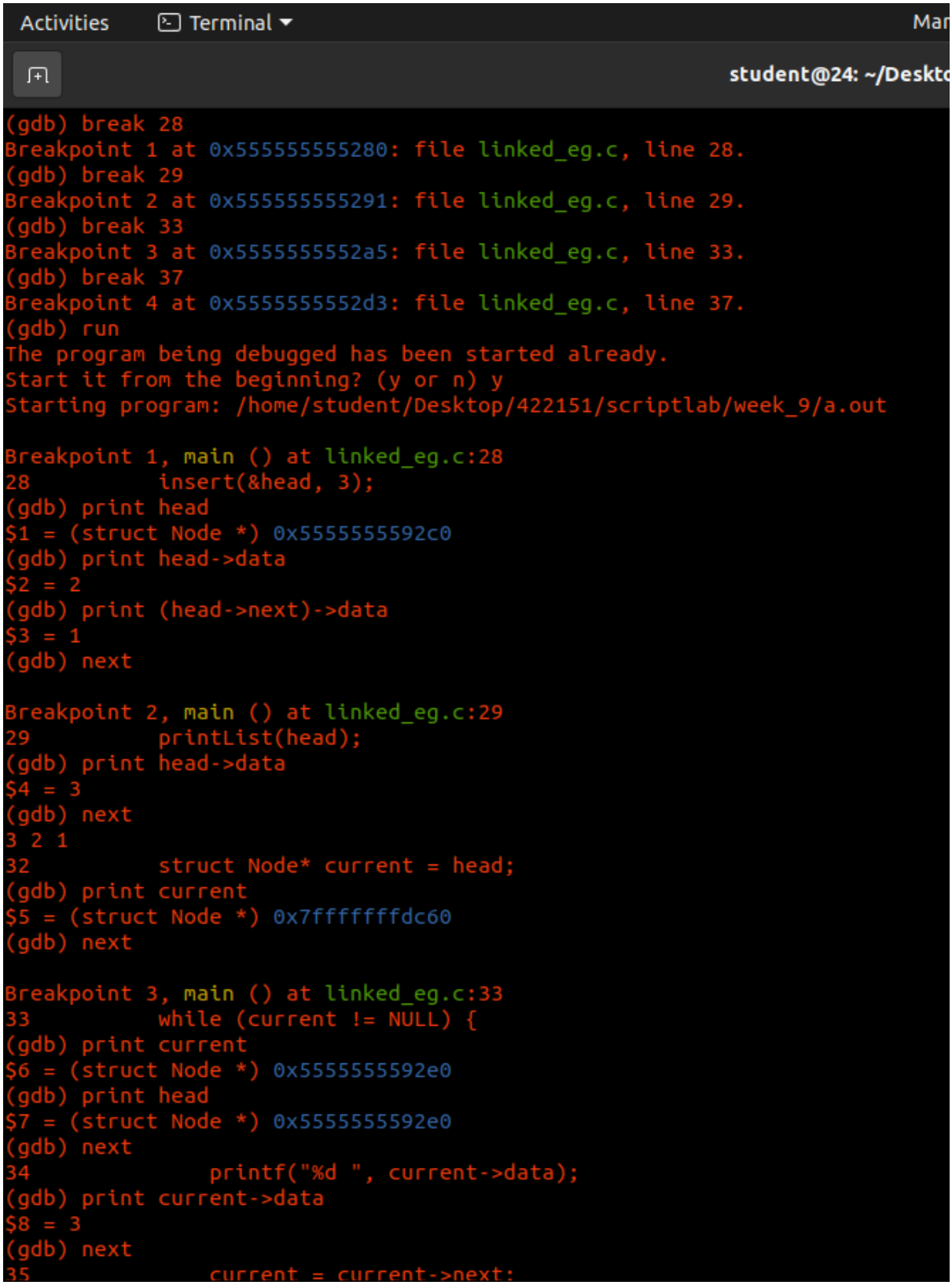
```
Activities   Terminal ▾                                          student@

13                          i++;
(gdb) continue
Continuing.
factorial of 5 is 1
factorial of 5 is 1
factorial of 5 is 1
[Inferior 1 (process 6160) exited normally]
(gdb) disassemble main
Dump of assembler code for function main:
   0x0000555555555189 <+0>:     endbr64
   0x000055555555518d <+4>:     push   %rbp
   0x000055555555518e <+5>:     mov    %rsp,%rbp
   0x0000555555555191 <+8>:     sub    $0x20,%rsp
   0x0000555555555195 <+12>:    mov    %fs:0x28,%rax
   0x000055555555519e <+21>:    mov    %rax,-0x8(%rbp)
   0x00005555555551a2 <+25>:    xor    %eax,%eax
   0x00005555555551a4 <+27>:    movq   $0x1,-0x10(%rbp)
   0x00005555555551ac <+35>:    lea    0xe55(%rip),%rdi        # 0x555555556008
   0x00005555555551b3 <+42>:    mov    $0x0,%eax
   0x00005555555551b8 <+47>:    callq  0x555555555080 <printf@plt>
   0x00005555555551bd <+52>:    lea    -0x18(%rbp),%rax
   0x00005555555551c1 <+56>:    mov    %rax,%rsi
   0x00005555555551c4 <+59>:    lea    0xe57(%rip),%rdi        # 0x555555556022
   0x00005555555551cb <+66>:    mov    $0x0,%eax
   0x00005555555551d0 <+71>:    callq  0x555555555090 <__isoc99_scanf@plt>
   0x00005555555551d5 <+76>:    mov    -0x18(%rbp),%eax
   0x00005555555551d8 <+79>:    test   %eax,%eax
   0x00005555555551da <+81>:    jns    0x5555555551ef <main+102>
   0x00005555555551dc <+83>:    lea    0xe45(%rip),%rdi        # 0x555555556028
   0x00005555555551e3 <+90>:    mov    $0x0,%eax
   0x00005555555551e8 <+95>:    callq  0x555555555080 <printf@plt>
   0x00005555555551ed <+100>:   jmp    0x55555555521e <main+149>
   0x00005555555551ef <+102>:   movl   $0x1,-0x14(%rbp)
   0x00005555555551f6 <+109>:   jmp    0x555555555216 <main+141>
   0x00005555555551f8 <+111>:   addl   $0x1,-0x14(%rbp)
   0x00005555555551fc <+115>:   mov    -0x18(%rbp),%eax
   0x00005555555551ff <+118>:   mov    -0x10(%rbp),%rdx
   0x0000555555555203 <+122>:   mov    %eax,%esi
   0x0000555555555205 <+124>:   lea    0xe52(%rip),%rdi        # 0x55555555605e
   0x000055555555520c <+131>:   mov    $0x0,%eax
   0x0000555555555211 <+136>:   callq  0x555555555080 <printf@plt>
   0x0000555555555216 <+141>:   mov    -0x18(%rbp),%eax
   0x0000555555555219 <+144>:   cmp    %eax,-0x14(%rbp)
   0x000055555555521c <+147>:   jle    0x5555555551f8 <main+111>
   0x000055555555521e <+149>:   mov    $0x0,%eax
   0x0000555555555223 <+154>:   mov    -0x8(%rbp),%rcx
   0x0000555555555227 <+158>:   xor    %fs:0x28,%rcx
   0x0000555555555230 <+167>:   je     0x555555555237 <main+174>
   0x0000555555555232 <+169>:   callq  0x555555555070 <__stack_chk_fail@plt>
   0x0000555555555237 <+174>:   leaveq
   0x0000555555555238 <+175>:   retq
End of assembler dump.
(gdb) quit
(base) student@24:~/Desktop/422151/scriptlab/week_9$
```

linked_eg.c:

```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

void insert(struct Node** head_ref, int new_data) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

void printList(struct Node* node) {
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
    printf("\n");
}

int main() {
    struct Node* head = NULL;
    insert(&head, 1);
    insert(&head, 2);
    insert(&head, 3);
    printList(head);

    // Accessing beyond the end of the list
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("%d",current->data);
    printf("\n");

    return 0;
}
```

Output:


Activities
Terminal
Mar 13 16:38
student@24: ~/Desktop/422151
(base) student@24:~/Desktop/422151/scriptlab/week_9$ gcc -g linked_eg.c
(base) student@24:~/Desktop/422151/scriptlab/week_9$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) run
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out
3 2 1
Program received signal SIGSEGV, Segmentation fault.
0x00005555555552d7 in main () at linked_eg.c:37
37 printf("%d",current->data);
(gdb) list
32 struct Node* current = head;
33 while (current != NULL) {
34 printf("%d ", current->data);
35 current = current->next;
36 }
37 printf("%d",current->data);
38 printf("\n");
39
40 return 0;
41 }
(gdb) list
Line number 42 out of range; linked_eg.c has 41 lines.
(gdb) list
Line number 42 out of range; linked_eg.c has 41 lines.
(gdb) break 28
Breakpoint 1 at 0x555555555280: file linked_eg.c, line 28.
(gdb) break 29
Breakpoint 2 at 0x555555555291: file linked_eg.c, line 29.
(gdb) break 33
Breakpoint 3 at 0x5555555552a5: file linked_eg.c, line 33.


Activities
Terminal
Mar
student@24: ~/Deskto
(gdb) break 28
Breakpoint 1 at 0x555555555280: file linked_eg.c, line 28.
(gdb) break 29
Breakpoint 2 at 0x555555555291: file linked_eg.c, line 29.
(gdb) break 33
Breakpoint 3 at 0x5555555552a5: file linked_eg.c, line 33.
(gdb) break 37
Breakpoint 4 at 0x5555555552d3: file linked_eg.c, line 37.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out
Breakpoint 1, main () at linked_eg.c:28
28 insert(&head, 3);
(gdb) print head
$1 = (struct Node *) 0x5555555592c0
(gdb) print head->data
$2 = 2
(gdb) print (head->next)->data
$3 = 1
(gdb) next
Breakpoint 2, main () at linked_eg.c:29
29 printList(head);
(gdb) print head->data
$4 = 3
(gdb) next
3 2 1
32 struct Node* current = head;
(gdb) print current
$5 = (struct Node *) 0x7fffffffdc60
(gdb) next
Breakpoint 3, main () at linked_eg.c:33
33 while (current != NULL) {
(gdb) print current
$6 = (struct Node *) 0x5555555592e0
(gdb) print head
$7 = (struct Node *) 0x5555555592e0
(gdb) next
34 printf("%d ", current->data);
(gdb) print current->data
$8 = 3
(gdb) next
35 current = current->next;

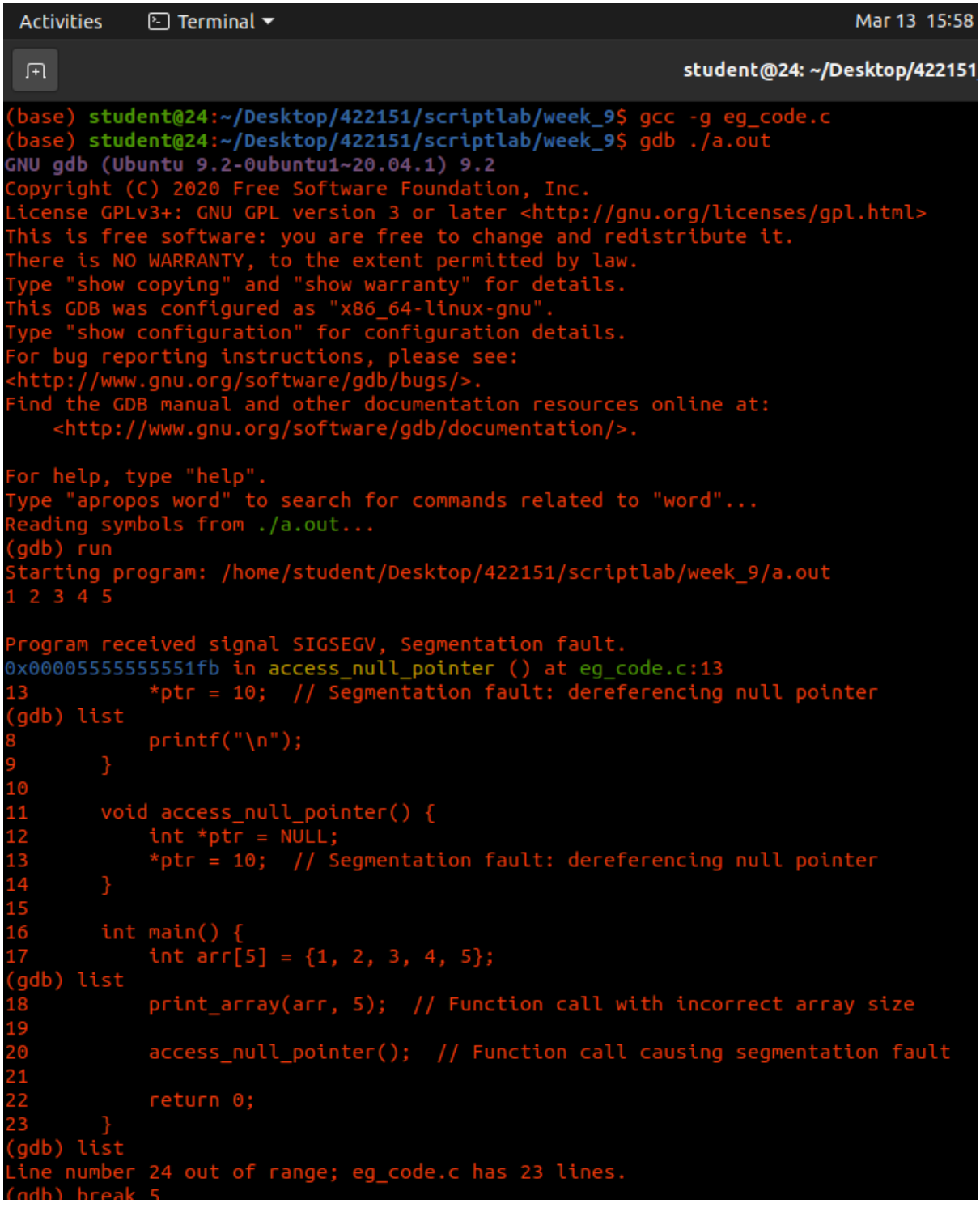
```
Breakpoint 3, main () at linked_eg.c:33
33          while (current != NULL) {
(gdb) print current
$6 = (struct Node *) 0x5555555592e0
(gdb) print head
$7 = (struct Node *) 0x5555555592e0
(gdb) next
34              printf("%d ", current->data);
(gdb) print current->data
$8 = 3
(gdb) next
35              current = current->next;
(gdb) print current->data
$9 = 3
(gdb) next
33          while (current != NULL) {
(gdb) print current->data
$10 = 2
(gdb) next
34              printf("%d ", current->data);
(gdb) print current->data
$11 = 2
(gdb) next
35              current = current->next;
(gdb) print current->data
$12 = 2
(gdb) next
33          while (current != NULL) {
(gdb) print current->data
$13 = 1
(gdb) next
34              printf("%d ", current->data);
(gdb) next
35              current = current->next;
(gdb) next
33          while (current != NULL) {
(gdb) next

Breakpoint 4, main () at linked_eg.c:37
37          printf("%d",current->data);
(gdb) print current
$14 = (struct Node *) 0x0
(gdb) print current->data
Cannot access memory at address 0x0
(gdb) next
```

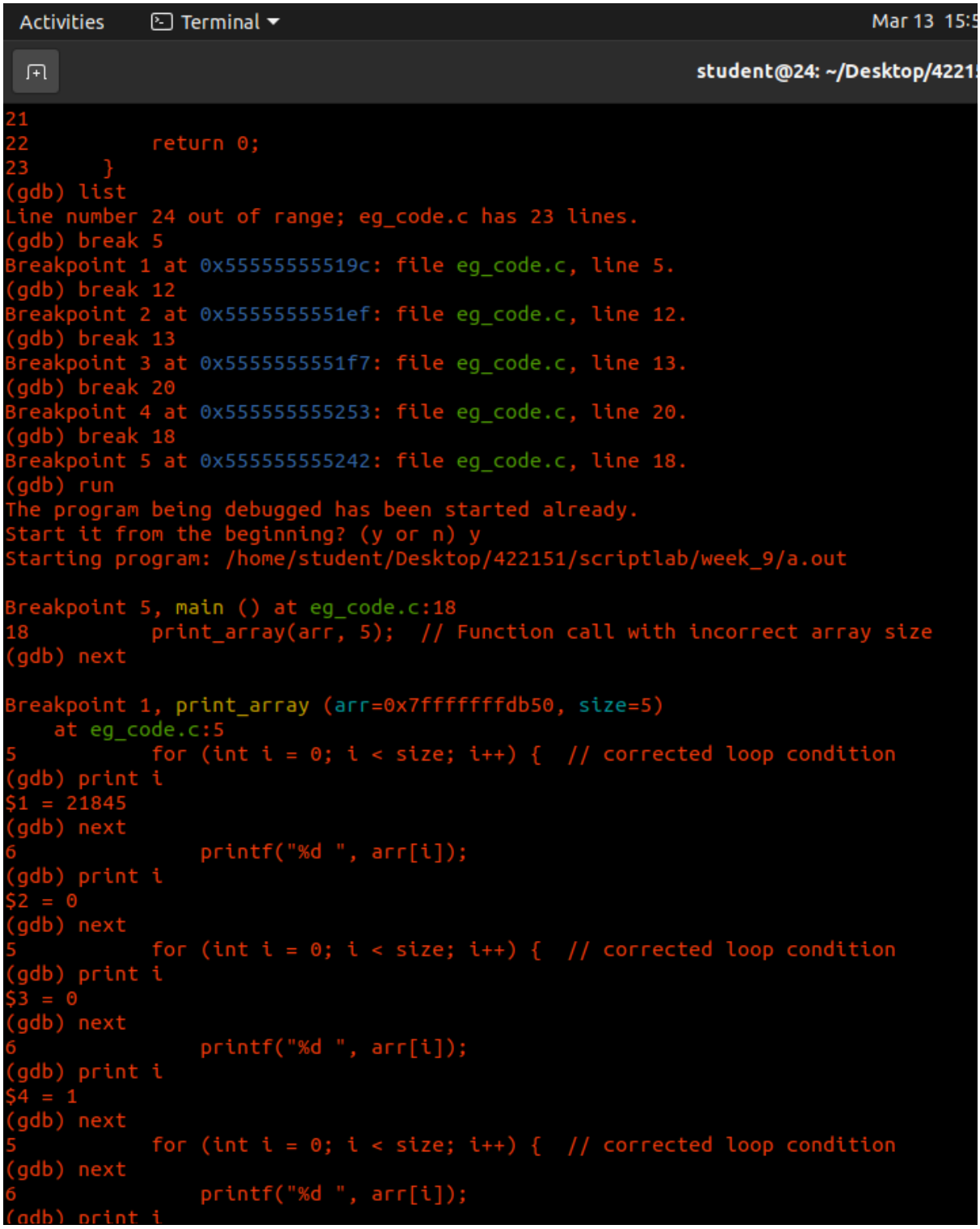
```
Breakpoint 4, main () at linked_eg.c:37
37          printf("%d",current->data);
(gdb) print current
$14 = (struct Node *) 0x0
(gdb) print current->data
Cannot access memory at address 0x0
(gdb) next

Program received signal SIGSEGV, Segmentation fault.
0x00005555555552d7 in main () at linked_eg.c:37
37          printf("%d",current->data);
(gdb) continue
Continuing.

Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb)
```

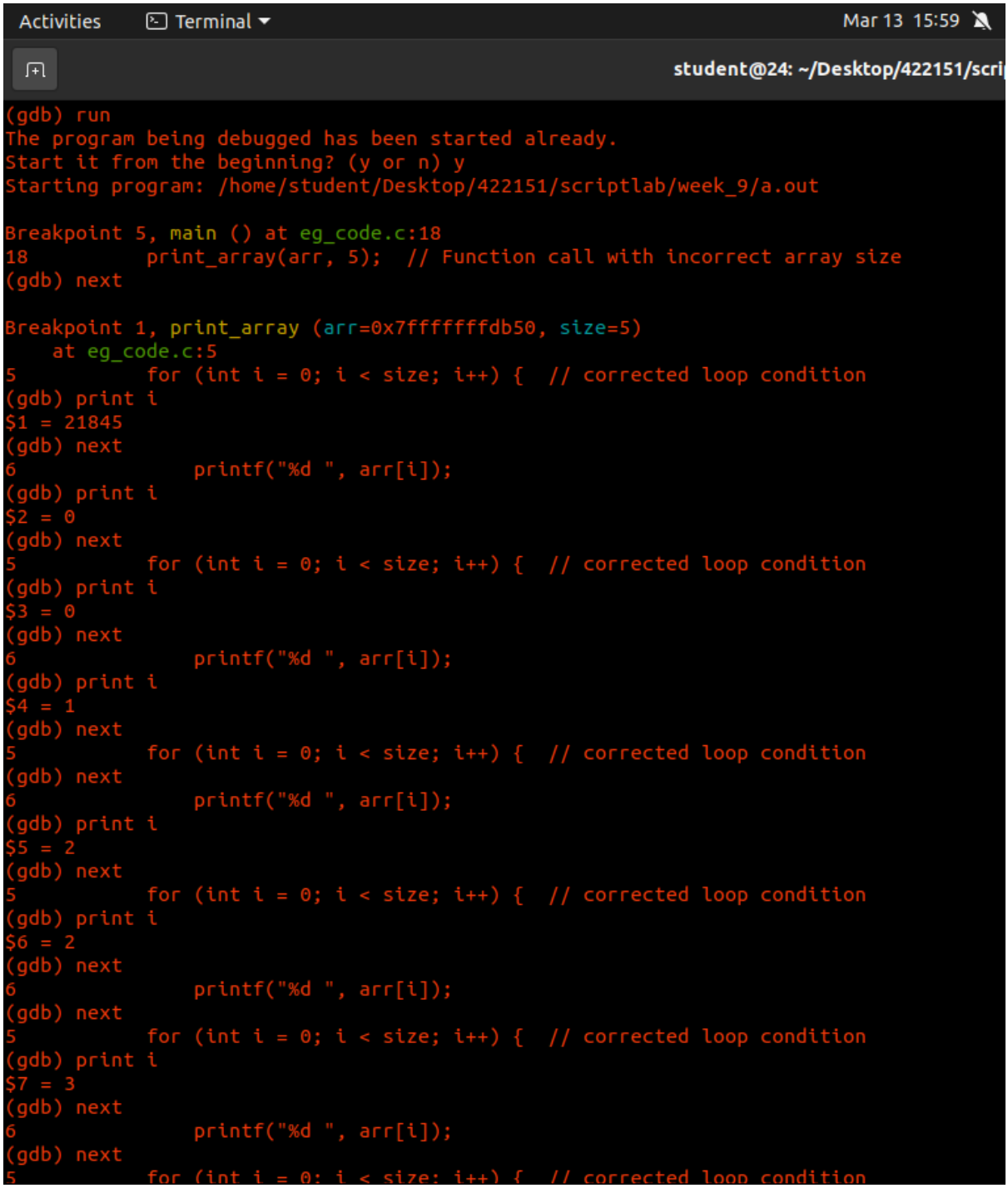
eg_code.c

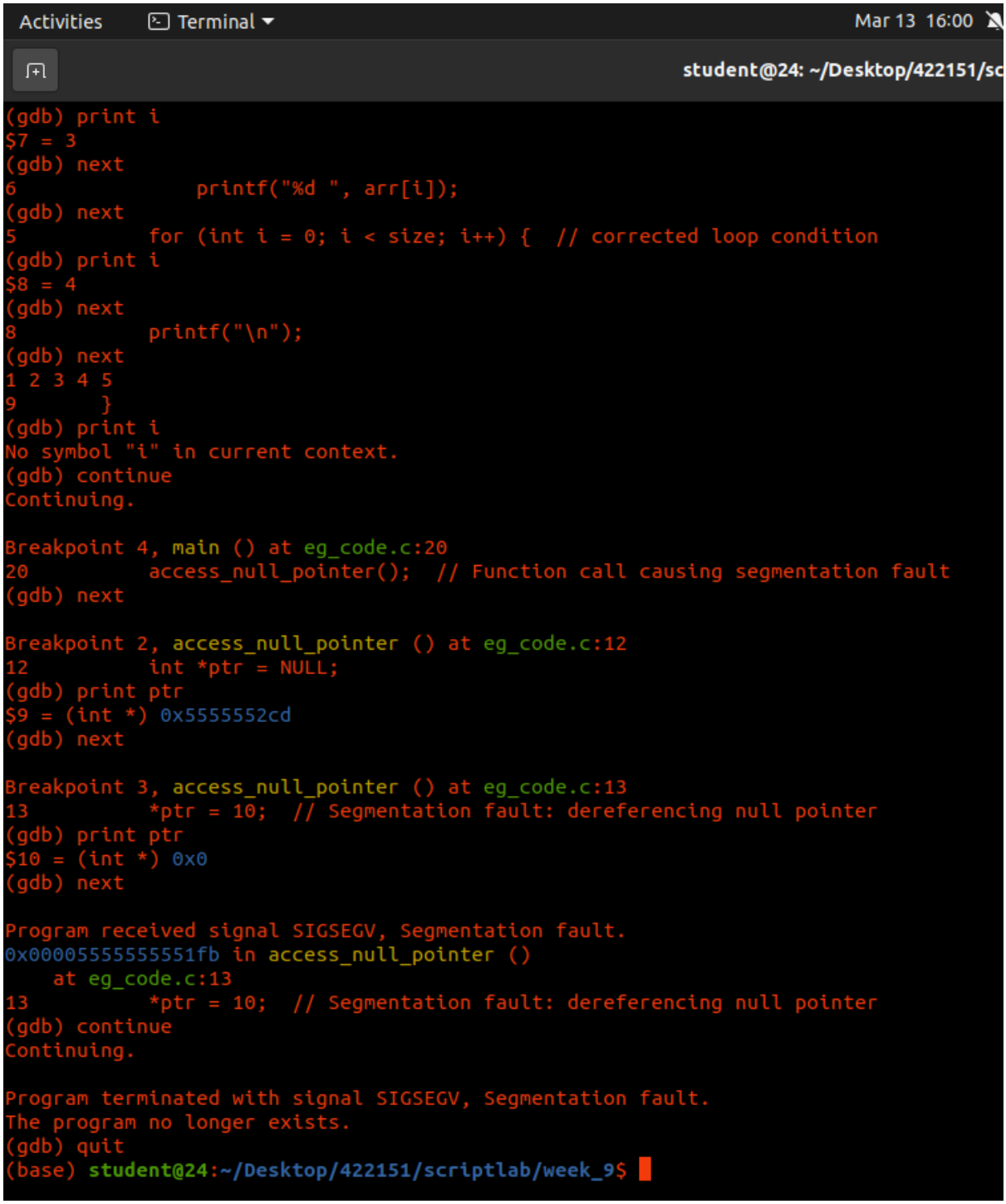
```c
#include <stdio.h>
#include <stdlib.h>

void print_array(int arr[], int size) {
    for (int i = 0; i < size; i++) {  // corrected loop condition
        printf("%d ", arr[i]);
    }
    printf("\n");
}

void access_null_pointer() {
    int *ptr = NULL;
    *ptr = 10;  // Segmentation fault: dereferencing null pointer
}

int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    print_array(arr, 5);  // Function call with incorrect array size

    access_null_pointer();  // Function call causing segmentation fault

    return 0;
}
```

Output


Activities
Terminal
Mar 13 15:58
student@24: ~/Desktop/422151
(base) student@24:~/Desktop/422151/scriptlab/week_9$ gcc -g eg_code.c
(base) student@24:~/Desktop/422151/scriptlab/week_9$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./a.out...
(gdb) run
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out
1 2 3 4 5
Program received signal SIGSEGV, Segmentation fault.
0x00005555555551fb in access_null_pointer () at eg_code.c:13
13 *ptr = 10; // Segmentation fault: dereferencing null pointer
(gdb) list
8 printf("\n");
9 }
10
11 void access_null_pointer() {
12 int *ptr = NULL;
13 *ptr = 10; // Segmentation fault: dereferencing null pointer
14 }
15
16 int main() {
17 int arr[5] = {1, 2, 3, 4, 5};
(gdb) list
18 print_array(arr, 5); // Function call with incorrect array size
19
20 access_null_pointer(); // Function call causing segmentation fault
21
22 return 0;
23 }
(gdb) list
Line number 24 out of range; eg_code.c has 23 lines.
(gdb) break 5


Activities
Terminal
Mar 13 15:5
student@24: ~/Desktop/4221
21
22 return 0;
23 }
(gdb) list
Line number 24 out of range; eg_code.c has 23 lines.
(gdb) break 5
Breakpoint 1 at 0x55555555519c: file eg_code.c, line 5.
(gdb) break 12
Breakpoint 2 at 0x5555555551ef: file eg_code.c, line 12.
(gdb) break 13
Breakpoint 3 at 0x5555555551f7: file eg_code.c, line 13.
(gdb) break 20
Breakpoint 4 at 0x555555555253: file eg_code.c, line 20.
(gdb) break 18
Breakpoint 5 at 0x555555555242: file eg_code.c, line 18.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out
Breakpoint 5, main () at eg_code.c:18
18 print_array(arr, 5); // Function call with incorrect array size
(gdb) next
Breakpoint 1, print_array (arr=0x7fffffffdb50, size=5)
at eg_code.c:5
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) print i
$1 = 21845
(gdb) next
6 printf("%d ", arr[i]);
(gdb) print i
$2 = 0
(gdb) next
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) print i
$3 = 0
(gdb) next
6 printf("%d ", arr[i]);
(gdb) print i
$4 = 1
(gdb) next
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) next
6 printf("%d ", arr[i]);
(gdb) print i


Activities
Terminal
Mar 13 15:59
student@24: ~/Desktop/422151/scri
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/student/Desktop/422151/scriptlab/week_9/a.out
Breakpoint 5, main () at eg_code.c:18
18 print_array(arr, 5); // Function call with incorrect array size
(gdb) next
Breakpoint 1, print_array (arr=0x7fffffffdb50, size=5)
at eg_code.c:5
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) print i
$1 = 21845
(gdb) next
6 printf("%d ", arr[i]);
(gdb) print i
$2 = 0
(gdb) next
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) print i
$3 = 0
(gdb) next
6 printf("%d ", arr[i]);
(gdb) print i
$4 = 1
(gdb) next
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) next
6 printf("%d ", arr[i]);
(gdb) print i
$5 = 2
(gdb) next
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) print i
$6 = 2
(gdb) next
6 printf("%d ", arr[i]);
(gdb) next
5 for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) print i
$7 = 3
(gdb) next
6 printf("%d ", arr[i]);
(gdb) next
5 for (int i = 0; i < size; i++) { // corrected loop condition


Activities
Terminal
Mar 13 16:00
student@24: ~/Desktop/422151/sc
(gdb) print i
$7 = 3
(gdb) next
6   printf("%d ", arr[i]);
(gdb) next
5   for (int i = 0; i < size; i++) { // corrected loop condition
(gdb) print i
$8 = 4
(gdb) next
8   printf("\n");
(gdb) next
1 2 3 4 5
9   }
(gdb) print i
No symbol "i" in current context.
(gdb) continue
Continuing.
Breakpoint 4, main () at eg_code.c:20
20   access_null_pointer(); // Function call causing segmentation fault
(gdb) next
Breakpoint 2, access_null_pointer () at eg_code.c:12
12   int *ptr = NULL;
(gdb) print ptr
$9 = (int *) 0x5555552cd
(gdb) next
Breakpoint 3, access_null_pointer () at eg_code.c:13
13   *ptr = 10; // Segmentation fault: dereferencing null pointer
(gdb) print ptr
$10 = (int *) 0x0
(gdb) next
Program received signal SIGSEGV, Segmentation fault.
0x00005555555551fb in access_null_pointer ()
    at eg_code.c:13
13   *ptr = 10; // Segmentation fault: dereferencing null pointer
(gdb) continue
Continuing.
Program terminated with signal SIGSEGV, Segmentation fault.
The program no longer exists.
(gdb) quit
(base) student@24:~/Desktop/422151/scriptlab/week_9$