ROLL NO : 422176
NAME : KOTA VENKATA CHARAN TEJA
SECTION:A
QUESTION:
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.

Note:
1. Include multiple breakpoints while debugging
2. Upload your submission in a format consistent with the example provided in the material

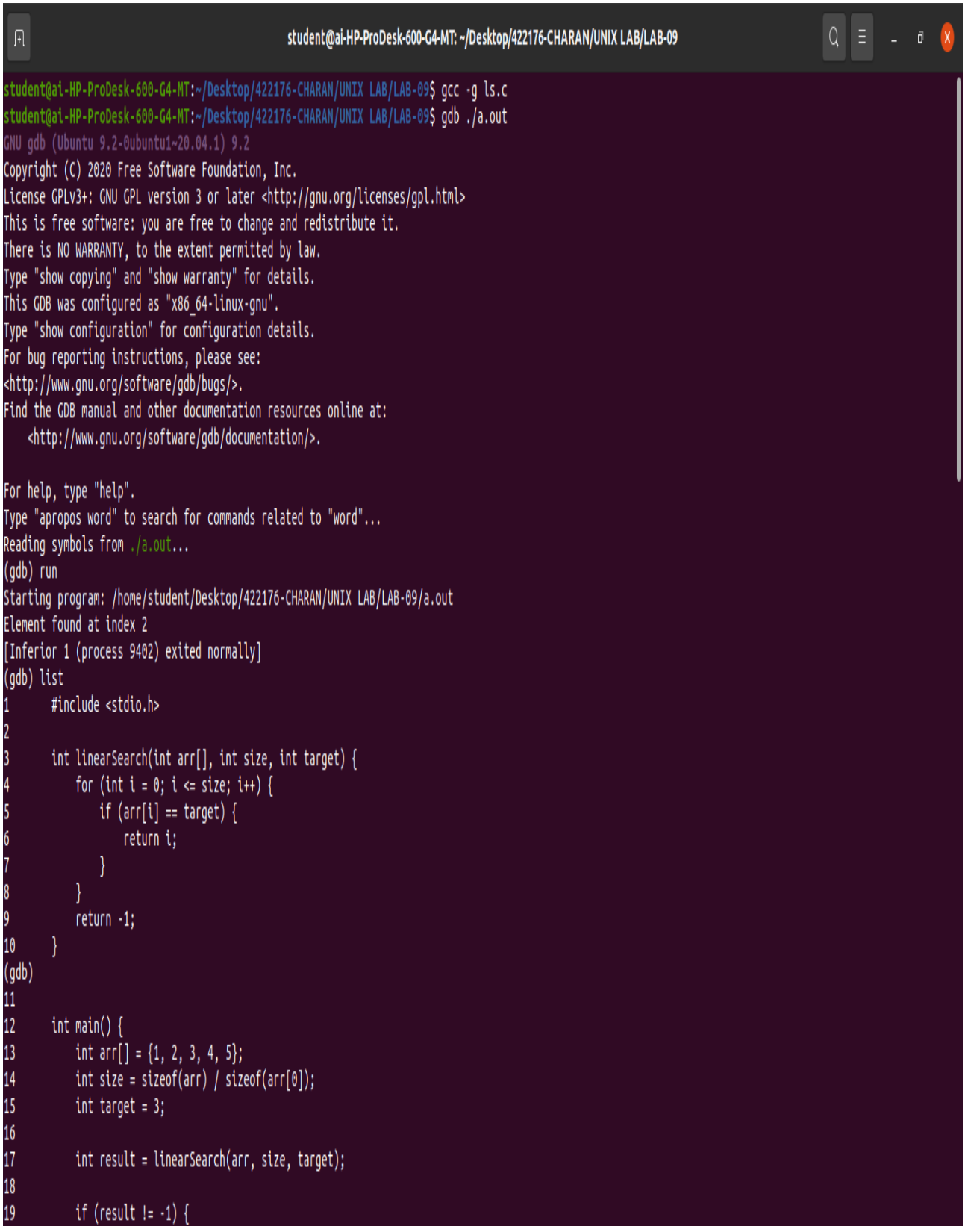
CODE:

WITH MULTIPLE BREAK POINTS

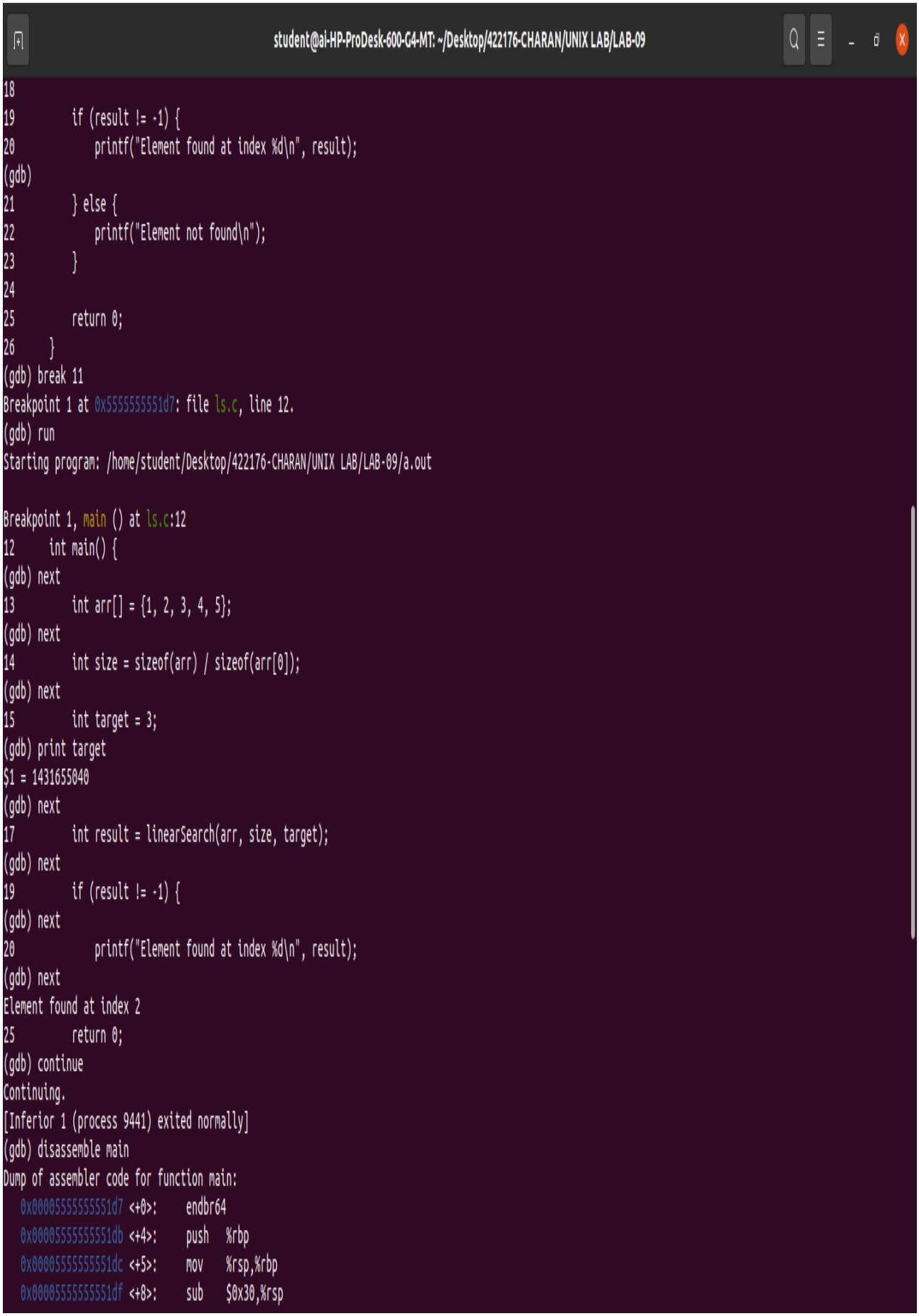
ls.c

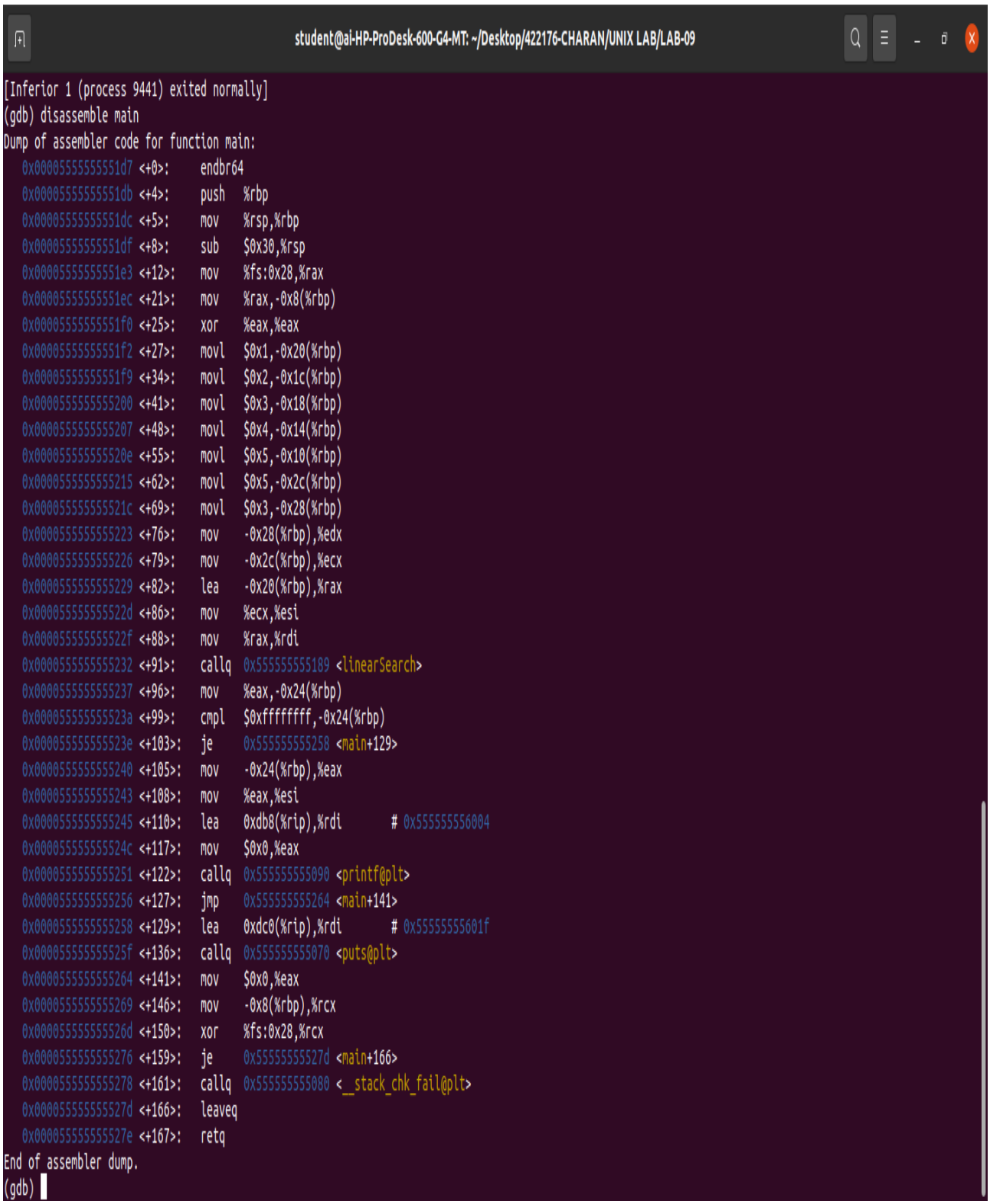
```
#include <stdio.h>

int linearSearch(int arr[], int size, int target) {
    for (int i = 0; i <= size; i++) {
        if (arr[i] == target) {
            return i;
        }
    }
    return -1;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int target = 3;

    int result = linearSearch(arr, size, target);

    if (result != -1) {
        printf("Element found at index %d\n", result);
    } else {
        printf("Element not found\n");
    }

    return 0;
}
```

DEBUGGING LS.C


student@ai-HP-ProDesk-600-G4-MT: ~/Desktop/422176-CHARAN/UNIX LAB/LAB-09
student@ai-HP-ProDesk-600-G4-MT:~/Desktop/422176-CHARAN/UNIX LAB/LAB-09$ gcc -g ls.c
student@ai-HP-ProDesk-600-G4-MT:~/Desktop/422176-CHARAN/UNIX LAB/LAB-09$ 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/422176-CHARAN/UNIX LAB/LAB-09/a.out
Element found at index 2
[Inferior 1 (process 9402) exited normally]
(gdb) list
1 #include <stdio.h>
2
3 int linearSearch(int arr[], int size, int target) {
4 for (int i = 0; i <= size; i++) {
5 if (arr[i] == target) {
6 return i;
7 }
8 }
9 return -1;
10 }
(gdb)
11
12 int main() {
13 int arr[] = {1, 2, 3, 4, 5};
14 int size = sizeof(arr) / sizeof(arr[0]);
15 int target = 3;
16
17 int result = linearSearch(arr, size, target);
18
19 if (result != -1) {

•


student@ai-HP-ProDesk-600-G4-MT: ~/Desktop/422176-CHARAN/UNIX LAB/LAB-09
18
19 if (result != -1) {
20 printf("Element found at index %d\n", result);
(gdb)
21 } else {
22 printf("Element not found\n");
23 }
24
25 return 0;
26 }
(gdb) break 11
Breakpoint 1 at 0x555555551d7: file ls.c, line 12.
(gdb) run
Starting program: /home/student/Desktop/422176-CHARAN/UNIX LAB/LAB-09/a.out
Breakpoint 1, main () at ls.c:12
12 int main() {
(gdb) next
13 int arr[] = {1, 2, 3, 4, 5};
(gdb) next
14 int size = sizeof(arr) / sizeof(arr[0]);
(gdb) next
15 int target = 3;
(gdb) print target
$1 = 1431655040
(gdb) next
17 int result = linearSearch(arr, size, target);
(gdb) next
19 if (result != -1) {
(gdb) next
20 printf("Element found at index %d\n", result);
(gdb) next
Element found at index 2
25 return 0;
(gdb) continue
Continuing.
[Inferior 1 (process 9441) exited normally]
(gdb) disassemble main
Dump of assembler code for function main:
0x00005555555551d7 <+0>: endbr64
0x00005555555551db <+4>: push %rbp
0x00005555555551dc <+5>: mov %rsp,%rbp
0x00005555555551df <+8>: sub $0x30,%rsp


student@ai-HP-ProDesk-600-G4-MT: ~/Desktop/422176-CHARAN/UNIX LAB/LAB-09
[Inferior 1 (process 9441) exited normally]
(gdb) disassemble main
Dump of assembler code for function main:
0x00005555555551d7 <+0>: endbr64
0x00005555555551db <+4>: push %rbp
0x00005555555551dc <+5>: mov %rsp,%rbp
0x00005555555551df <+8>: sub $0x30,%rsp
0x00005555555551e3 <+12>: mov %fs:0x28,%rax
0x00005555555551ec <+21>: mov %rax,-0x8(%rbp)
0x00005555555551f0 <+25>: xor %eax,%eax
0x00005555555551f2 <+27>: movl $0x1,-0x20(%rbp)
0x00005555555551f9 <+34>: movl $0x2,-0x1c(%rbp)
0x0000555555555200 <+41>: movl $0x3,-0x18(%rbp)
0x0000555555555207 <+48>: movl $0x4,-0x14(%rbp)
0x000055555555520e <+55>: movl $0x5,-0x10(%rbp)
0x0000555555555215 <+62>: movl $0x5,-0x2c(%rbp)
0x000055555555521c <+69>: movl $0x3,-0x28(%rbp)
0x0000555555555223 <+76>: mov -0x28(%rbp),%edx
0x0000555555555226 <+79>: mov -0x2c(%rbp),%ecx
0x0000555555555229 <+82>: lea -0x20(%rbp),%rax
0x000055555555522d <+86>: mov %ecx,%esi
0x000055555555522f <+88>: mov %rax,%rdi
0x0000555555555232 <+91>: callq 0x555555555189 <linearSearch>
0x0000555555555237 <+96>: mov %eax,-0x24(%rbp)
0x000055555555523a <+99>: cmpl $0xffffffff,-0x24(%rbp)
0x000055555555523e <+103>: je 0x555555555258 <main+129>
0x0000555555555240 <+105>: mov -0x24(%rbp),%eax
0x0000555555555243 <+108>: mov %eax,%esi
0x0000555555555245 <+110>: lea 0xdb8(%rip),%rdi # 0x555555556004
0x000055555555524c <+117>: mov $0x0,%eax
0x0000555555555251 <+122>: callq 0x555555555090 <printf@plt>
0x0000555555555256 <+127>: jmp 0x555555555264 <main+141>
0x0000555555555258 <+129>: lea 0xdc0(%rip),%rdi # 0x55555555601f
0x000055555555525f <+136>: callq 0x555555555070 <puts@plt>
0x0000555555555264 <+141>: mov $0x0,%eax
0x0000555555555269 <+146>: mov -0x8(%rbp),%rcx
0x000055555555526d <+150>: xor %fs:0x28,%rcx
0x0000555555555276 <+159>: je 0x55555555527d <main+166>
0x0000555555555278 <+161>: callq 0x555555555080 <__stack_chk_fail@plt>
0x000055555555527d <+166>: leaveq
0x000055555555527e <+167>: retq
End of assembler dump.
(gdb)