UNIX ASSIGNMENT – 4

NAME:C.PAVITHRA

ROLL NO:422127

SECTION: 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.

1.factorial:

Code:

```
#include <stdio.h>
int main() {
    int num;
    unsigned long long factorial = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    if (num < 0) {
        printf("Factorial of a negative number is not defined.\n");
    } else {
        int i = 1;

        while(i<=num) {
            factorial *= i;
            i++;
            printf("Factorial of %d is %llu\n", num, factorial);
        }
    }
    return 0;
}
```

Gdb:

```
student@ai-HP-ProDesk-600-G4-MT:~$ cd Desktop/422127_unixlab
student@ai-HP-ProDesk-600-G4-MT:~/Desktop/422127_unixlab$ gcc -g exp.c
student@ai-HP-ProDesk-600-G4-MT:~/Desktop/422127_unixlab$ gdb ./a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 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/422127_unixlab/a.out
Enter a positive integer: 5
Factorial of 5 is 1
Factorial of 5 is 2
Factorial of 5 is 6
Factorial of 5 is 24
Factorial of 5 is 120
[Inferior 1 (process 7186) exited normally]
(gdb) list
1       #include <stdio.h>
2       int main() {
3           int num;
4           unsigned long long factorial = 1;
5           printf("Enter a positive integer: ");
6           scanf("%d", &num);
7           if (num < 0) {
8               printf("Factorial of a negative number is not defined.\n");
9           } else {
10              int i = 1;
(gdb)
11
12              while(i<=num) {
13                  factorial *= i;
```

```
11
12              while(i<=num) {
13                  factorial *= i;
14                  i++;
15                  printf("Factorial of %d is %llu\n", num, factorial);
16              }
17          }
18          return 0;
19      }
20
(gdb)
Line number 21 out of range; exp.c has 20 lines.
(gdb) break 11
Breakpoint 1 at 0x555555555211: file exp.c, line 12.
(gdb) run
Starting program: /home/student/Desktop/422127_unixlab/a.out
Enter a positive integer: 5

Breakpoint 1, main () at exp.c:12
12              while(i<=num) {
(gdb) print i
$1 = 1
(gdb) print num
$2 = 5
(gdb) next
13                  factorial *= i;
(gdb) next
14                  i++;
(gdb) print factorial
$3 = 1
(gdb) next
15                  printf("Factorial of %d is %llu\n", num, factorial);
(gdb) print i
$4 = 2
(gdb) next
Factorial of 5 is 1
12              while(i<=num) {
(gdb) print i
$5 = 2
(gdb) print num
$6 = 5
(gdb) next
13                  factorial *= i;
```

```
$5 = 2
(gdb) print num
$6 = 5
(gdb) next
13                  factorial *= i;
(gdb) next
14                  i++;
(gdb) print i
$7 = 2
(gdb) next
15                  printf("Factorial of %d is %llu\n", num, factorial);
(gdb) next
Factorial of 5 is 2
12              while(i<=num) {
(gdb) continue
Continuing.
Factorial of 5 is 6
Factorial of 5 is 24
Factorial of 5 is 120
[Inferior 1 (process 7199) exited normally]
(gdb) disassemble main
Dump of assembler code for function main:
   0x00005555555551a9 <+0>:     endbr64
   0x00005555555551ad <+4>:     push   %rbp
   0x00005555555551ae <+5>:     mov    %rsp,%rbp
   0x00005555555551b1 <+8>:     sub    $0x20,%rsp
   0x00005555555551b5 <+12>:    mov    %fs:0x28,%rax
   0x00005555555551be <+21>:    mov    %rax,-0x8(%rbp)
   0x00005555555551c2 <+25>:    xor    %eax,%eax
   0x00005555555551c4 <+27>:    movq   $0x1,-0x10(%rbp)
   0x00005555555551cc <+35>:    lea    0xe35(%rip),%rdi        # 0x555555556008
   0x00005555555551d3 <+42>:    mov    $0x0,%eax
   0x00005555555551d8 <+47>:    callq  0x5555555550a0 <printf@plt>
   0x00005555555551dd <+52>:    lea    -0x18(%rbp),%rax
   0x00005555555551e1 <+56>:    mov    %rax,%rsi
   0x00005555555551e4 <+59>:    lea    0xe38(%rip),%rdi        # 0x555555556023
   0x00005555555551eb <+66>:    mov    $0x0,%eax
   0x00005555555551f0 <+71>:    callq  0x5555555550b0 <__isoc99_scanf@plt>
   0x00005555555551f5 <+76>:    mov    -0x18(%rbp),%eax
   0x00005555555551f8 <+79>:    test   %eax,%eax
   0x00005555555551fa <+81>:    jns    0x55555555520a <main+97>
   0x00005555555551fc <+83>:    lea    0xe25(%rip),%rdi        # 0x555555556028
   0x0000555555555203 <+90>:    callq  0x5555555555080 <puts@plt>
```

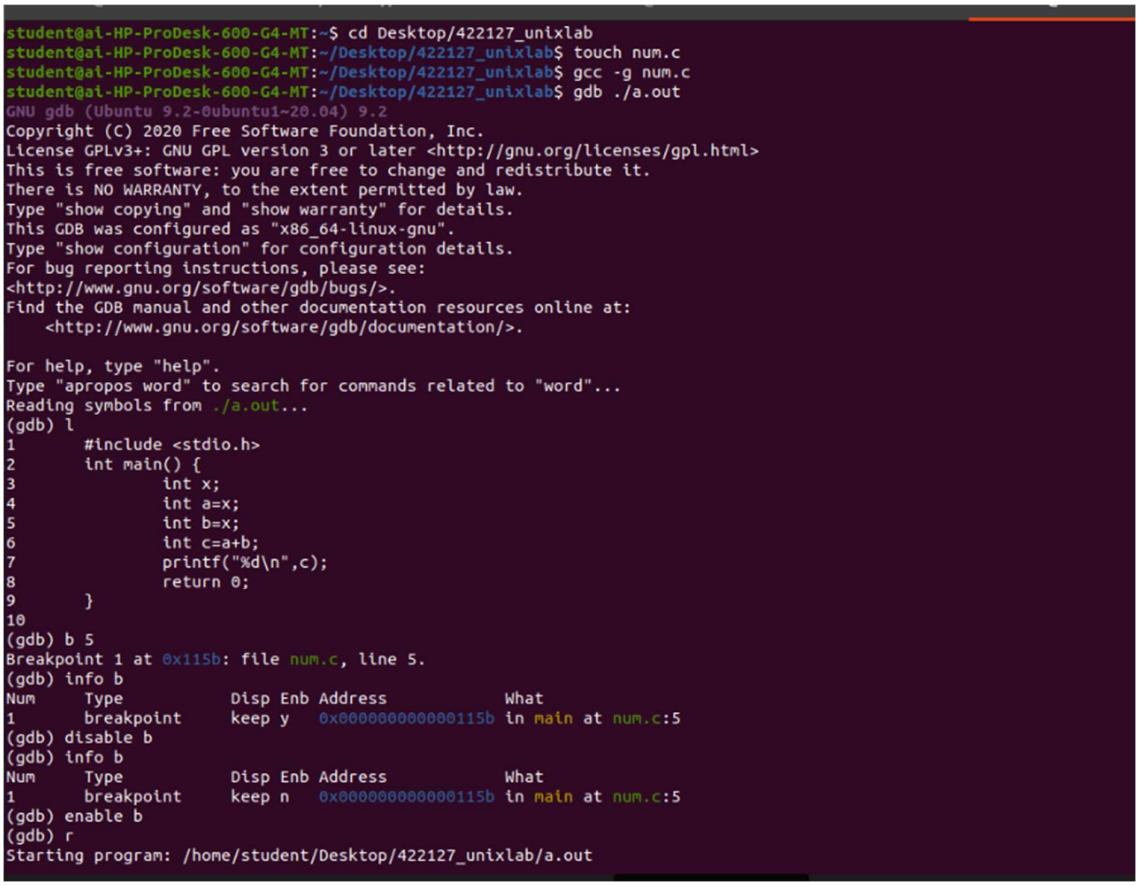
```
   0x00005555555551cc <+35>:    lea    0xe35(%rip),%rdi        # 0x555555556008
   0x00005555555551d3 <+42>:    mov    $0x0,%eax
   0x00005555555551d8 <+47>:    callq  0x5555555550a0 <printf@plt>
   0x00005555555551dd <+52>:    lea    -0x18(%rbp),%rax
   0x00005555555551e1 <+56>:    mov    %rax,%rsi
   0x00005555555551e4 <+59>:    lea    0xe38(%rip),%rdi        # 0x555555556023
   0x00005555555551eb <+66>:    mov    $0x0,%eax
   0x00005555555551f0 <+71>:    callq  0x5555555550b0 <__isoc99_scanf@plt>
   0x00005555555551f5 <+76>:    mov    -0x18(%rbp),%eax
   0x00005555555551f8 <+79>:    test   %eax,%eax
   0x00005555555551fa <+81>:    jns    0x55555555520a <main+97>
   0x00005555555551fc <+83>:    lea    0xe25(%rip),%rdi        # 0x555555556028
   0x0000555555555203 <+90>:    callq  0x5555555555080 <puts@plt>
   0x0000555555555208 <+95>:    jmp    0x55555555524a <main+161>
--Type <RET> for more, q to quit, c to continue without paging--
   0x000055555555520a <+97>:    movl   $0x1,-0x14(%rbp)
   0x0000555555555211 <+104>:   jmp    0x555555555242 <main+153>
   0x0000555555555213 <+106>:   mov    -0x14(%rbp),%eax
   0x0000555555555216 <+109>:   cltq
   0x0000555555555218 <+111>:   mov    -0x10(%rbp),%rdx
   0x000055555555521c <+115>:   imul   %rdx,%rax
   0x0000555555555220 <+119>:   mov    %rax,-0x10(%rbp)
   0x0000555555555224 <+123>:   addl   $0x1,-0x14(%rbp)
   0x0000555555555228 <+127>:   mov    -0x18(%rbp),%eax
   0x000055555555522b <+130>:   mov    -0x10(%rbp),%rdx
   0x000055555555522f <+134>:   mov    %eax,%esi
   0x0000555555555231 <+136>:   lea    0xe1f(%rip),%rdi        # 0x555555556057
   0x0000555555555238 <+143>:   mov    $0x0,%eax
   0x000055555555523d <+148>:   callq  0x5555555550a0 <printf@plt>
   0x0000555555555242 <+153>:   mov    -0x18(%rbp),%eax
   0x0000555555555245 <+156>:   cmp    %eax,-0x14(%rbp)
   0x0000555555555248 <+159>:   jle    0x555555555213 <main+106>
   0x000055555555524a <+161>:   mov    $0x0,%eax
   0x000055555555524f <+166>:   mov    -0x8(%rbp),%rcx
   0x0000555555555253 <+170>:   xor    %fs:0x28,%rcx
   0x000055555555525c <+179>:   je     0x555555555263 <main+186>
   0x000055555555525e <+181>:   callq  0x5555555555090 <__stack_chk_fail@plt>
   0x0000555555555263 <+186>:   leaveq
--Type <RET> for more, q to quit, c to continue without paging--
   0x0000555555555264 <+187>:   retq
End of assembler dump.
(gdb) quit
```

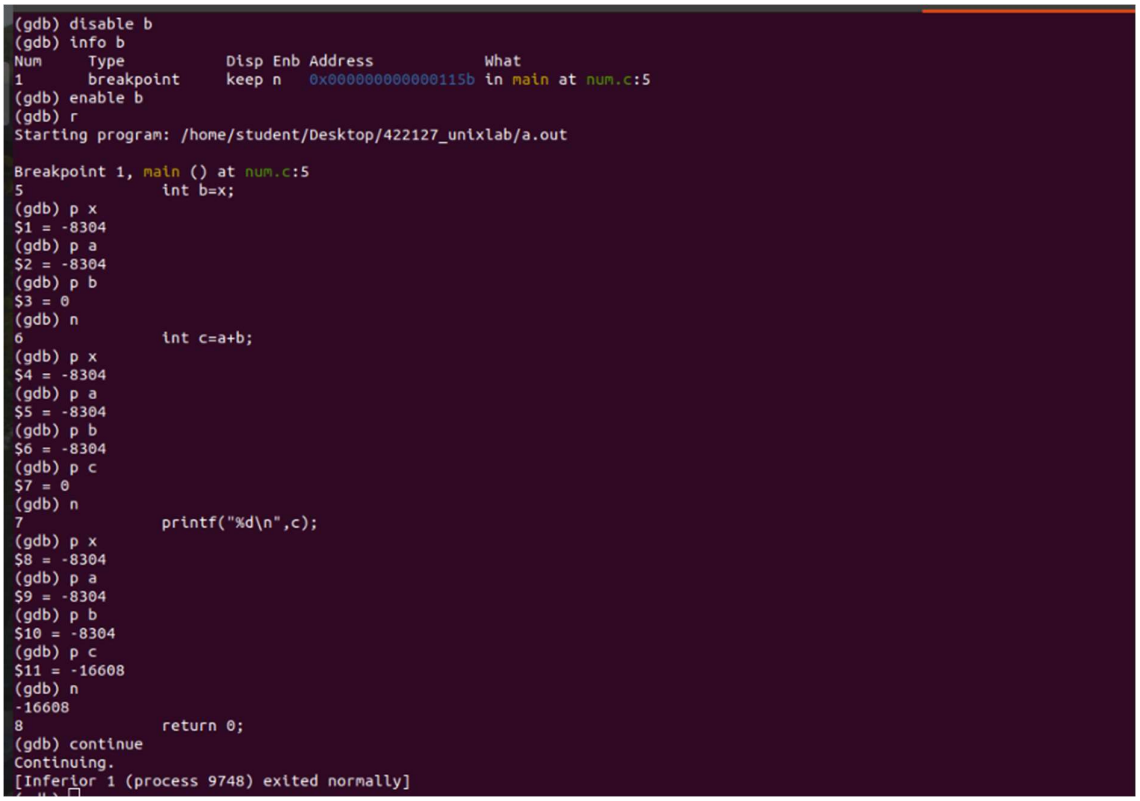
2.addition:

Code:

```
#include <stdio.h>

int main() {

        int x;

        int a = x;

        int b = x;

        int c = a+b;

        printf(“%d\n”,c);

        return 0;

}
```

Gdb:




(gdb) disable b
(gdb) info b
Num Type Disp Enb Address What
1 breakpoint keep n 0x000000000000115b in main at num.c:5
(gdb) enable b
(gdb) r
Starting program: /home/student/Desktop/422127_unixlab/a.out
Breakpoint 1, main () at num.c:5
5 int b=x;
(gdb) p x
$1 = -8304
(gdb) p a
$2 = -8304
(gdb) p b
$3 = 0
(gdb) n
6 int c=a+b;
(gdb) p x
$4 = -8304
(gdb) p a
$5 = -8304
(gdb) p b
$6 = -8304
(gdb) p c
$7 = 0
(gdb) n
7 printf("%d\n",c);
(gdb) p x
$8 = -8304
(gdb) p a
$9 = -8304
(gdb) p b
$10 = -8304
(gdb) p c
$11 = -16608
(gdb) n
-16608
8 return 0;
(gdb) continue
Continuing.
[Inferior 1 (process 9748) exited normally]