Bil 461 – Ödev 3

Fatih Furkan Has
141101024

1-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ gdb null
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 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 null...done.
(gdb) run
Starting program: /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/null

Program received signal SIGSEGV, Segmentation fault.
0x000055555555460a in main () at null.c:7
7         *pointer = 106071996;
(gdb) Quit
(gdb) quit
A debugging session is active.

        Inferior 1 [process 6676] will be killed.

Quit anyway? (y or n) y
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ ./null
Segmentation fault (core dumped)
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$
```

Programımızı normal çalıştırmaya çalıştığımızda “Segmentation fault” hatası almaktayız. Ancak gdb ile debug ettiğimizde görüldüğü gibi hatanın hangi satırda olduğunu, hangi değişkende olduğunu da ayrıntılı olarak görebilmekteyiz.

2-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ valgrind --leak-check=yes ./null
==10967== Memcheck, a memory error detector
==10967== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10967== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==10967== Command: ./null
==10967==
==10967== Invalid write of size 4
==10967==    at 0x10860A: main (null.c:7)
==10967==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
==10967==
==10967==
==10967== Process terminating with default action of signal 11 (SIGSEGV)
==10967==  Access not within mapped region at address 0x0
==10967==    at 0x10860A: main (null.c:7)
==10967==  If you believe this happened as a result of a stack
==10967==  overflow in your program's main thread (unlikely but
==10967==  possible), you can try to increase the size of the
==10967==  main thread stack using the --main-stacksize= flag.
==10967==  The main thread stack size used in this run was 8388608.
==10967==
==10967== HEAP SUMMARY:
==10967==     in use at exit: 0 bytes in 0 blocks
==10967==   total heap usage: 0 allocs, 0 frees, 0 bytes allocated
==10967==
==10967== All heap blocks were freed -- no leaks are possible
==10967==
==10967== For counts of detected and suppressed errors, rerun with: -v
==10967== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
Segmentation fault (core dumped)
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$
```

valgrind --leak-check=yes ./null komutuyla çalıştırdığımızda görüyoruz ki
“Invalid write of size 4
at 0x10860A: main (null.c:7)
Address 0x0 is not stack'd, malloc'd or (recently) free'd” yani atama yapmayaca çalıştığımız pointer için hafızada bir yer ayrılmamış bu yüzden atama yapılamamaktadır.

3-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ gcc soru3.c -o soru3
soru3.c: In function ‘main’:
soru3.c:5:20: warning: implicit declaration of function ‘malloc’ [-Wimplicit-function-declaration]
     int* pointer = malloc(50);
                    ^~~~~~
soru3.c:5:20: warning: incompatible implicit declaration of built-in function ‘malloc’
soru3.c:5:20: note: include ‘<stdlib.h>’ or provide a declaration of ‘malloc’
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ gdb soru3
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 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 soru3...(no debugging symbols found)...done.
(gdb) run
Starting program: /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru3
[Inferior 1 (process 12327) exited normally]
(gdb) quit
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ valgrind --leak-check=yes ./soru3
==12332== Memcheck, a memory error detector
==12332== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12332== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==12332== Command: ./soru3
==12332==
==12332==
==12332== HEAP SUMMARY:
==12332==     in use at exit: 50 bytes in 1 blocks
==12332==   total heap usage: 1 allocs, 0 frees, 50 bytes allocated
==12332==
==12332== 50 bytes in 1 blocks are definitely lost in loss record 1 of 1
==12332==    at 0x4C2FB6B: malloc (vg_replace_malloc.c:299)
==12332==    by 0x10865B: main (in /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru3)
==12332==
==12332== LEAK SUMMARY:
==12332==    definitely lost: 50 bytes in 1 blocks
==12332==    indirectly lost: 0 bytes in 0 blocks
==12332==      possibly lost: 0 bytes in 0 blocks
==12332==    still reachable: 0 bytes in 0 blocks
==12332==         suppressed: 0 bytes in 0 blocks
==12332==
==12332== For counts of detected and suppressed errors, rerun with: -v
==12332== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

gdb ile debug etmeye çalıştığımızda gdb herhangi bir problem bulamadı ve normal bir şekilde çalışıp sonlandığını gösterdi. Ancak valgrind ile baktığımızda bize “1 error” olduğunu belirtti buradaki hata 50 byte lik bir alan malloc ettik ancak bunun asla free yapılmaması.

4-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ gcc soru4.c -o soru4
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ ./soru4
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$
```

Programımızı çalıştırdığımızda hiçbir error veya warning almadım.

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ gcc soru4.c -o soru4
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ ./soru4
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ valgrind --leak-check=yes ./soru4
==12450== Memcheck, a memory error detector
==12450== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12450== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==12450== Command: ./soru4
==12450==
==12450== Invalid write of size 4
==12450==    at 0x10866A: main (in /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru4)
==12450==  Address 0x522d1d0 is 0 bytes after a block of size 400 alloc'd
==12450==    at 0x4C2FB6B: malloc (vg_replace_malloc.c:299)
==12450==    by 0x10865B: main (in /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru4)
==12450==
==12450==
==12450== HEAP SUMMARY:
==12450==     in use at exit: 400 bytes in 1 blocks
==12450==   total heap usage: 1 allocs, 0 frees, 400 bytes allocated
==12450==
==12450== 400 bytes in 1 blocks are definitely lost in loss record 1 of 1
==12450==    at 0x4C2FB6B: malloc (vg_replace_malloc.c:299)
==12450==    by 0x10865B: main (in /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru4)
==12450==
==12450== LEAK SUMMARY:
==12450==    definitely lost: 400 bytes in 1 blocks
==12450==    indirectly lost: 0 bytes in 0 blocks
==12450==      possibly lost: 0 bytes in 0 blocks
==12450==    still reachable: 0 bytes in 0 blocks
==12450==         suppressed: 0 bytes in 0 blocks
==12450==
==12450== For counts of detected and suppressed errors, rerun with: -v
==12450== ERROR SUMMARY: 2 errors from 2 contexts (suppressed: 0 from 0)
```

valgrind ile çalıştırdığımızda ise vize 2 hata olduğunu söyledi. İlk hata data[100] yapmamızla alakalı çünkü biz 100 integer alacak bir array için bir alan allocate ettik ancak 101. alana yazmaya çalıştık. İkinci hata ise program sonunda allocate edilen alanı free yapmamız. Programın çalışmasında herhangi bir sıkıntı yok program yanlış diyemeyiz ama 100 elemanlık bir arrayin 101. elemanına atama yapmak da yanlış bir program olduğunu söyleyebilir.

5-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ gcc soru5.c -o soru5
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ ./soru5
0
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3$ valgrind --leak-check=yes ./soru5
==12938== Memcheck, a memory error detector
==12938== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12938== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==12938== Command: ./soru5
==12938==
==12938== Invalid read of size 4
==12938==    at 0x10870A: main (in /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru5)
==12938==  Address 0x522d040 is 0 bytes inside a block of size 400 free'd
==12938==    at 0x4C30D18: free (vg_replace_malloc.c:530)
==12938==    by 0x108705: main (in /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru5)
==12938==  Block was alloc'd at
==12938==    at 0x4C2FB6B: malloc (vg_replace_malloc.c:299)
==12938==    by 0x1086EB: main (in /home/toor/Desktop/461 - Isletim Sistemleri/Odev3/soru5)
==12938==
1607
==12938==
==12938== HEAP SUMMARY:
==12938==     in use at exit: 0 bytes in 0 blocks
==12938==   total heap usage: 2 allocs, 2 frees, 1,424 bytes allocated
==12938==
==12938== All heap blocks were freed -- no leaks are possible
==12938==
==12938== For counts of detected and suppressed errors, rerun with: -v
==12938== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Array'i oluşturduktan sonra 0. eleman için 1607 değerini atadım. Daha sonra free(data) yaptım ve bu elemanı ekrana bastırmak istediğimde ekrana 0 değerini bastı.

6-)

Part 2:

1)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 4

ARG seed 4
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x0000069a (decimal 1690)
  Limit  : 316

Virtual Address Trace
  VA  0: 0x00000195 (decimal:  405) --> PA or segmentation violation?
  VA  1: 0x0000009e (decimal:  158) --> PA or segmentation violation?
  VA  2: 0x00000044 (decimal:   68) --> PA or segmentation violation?
  VA  3: 0x0000019b (decimal:  411) --> PA or segmentation violation?
  VA  4: 0x000003ab (decimal:  939) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple virtual address space of a given size.

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 5

ARG seed 5
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00002f79 (decimal 12153)
  Limit  : 415

Virtual Address Trace
  VA  0: 0x0000032e (decimal:  814) --> PA or segmentation violation?
  VA  1: 0x000003c5 (decimal:  965) --> PA or segmentation violation?
  VA  2: 0x000002f5 (decimal:  757) --> PA or segmentation violation?
  VA  3: 0x000003b0 (decimal:  944) --> PA or segmentation violation?
  VA  4: 0x0000001d (decimal:   29) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple virtual address space of a given size.

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 6

ARG seed 6
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x0000349a (decimal 13466)
  Limit  : 459

Virtual Address Trace
  VA  0: 0x000001f0 (decimal:  496) --> PA or segmentation violation?
  VA  1: 0x0000010b (decimal:  267) --> PA or segmentation violation?
  VA  2: 0x00000000 (decimal:    0) --> PA or segmentation violation?
  VA  3: 0x000002a6 (decimal:  678) --> PA or segmentation violation?
  VA  4: 0x000001e1 (decimal:  481) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple virtual address space of a given size.
```

Eğer decimal değerimiz limit değerimizden büyükse bir segmentation violation olmakta. Yani out of bound olmakta. Eğer limit değerimizden küçük ise decimal değer ile base'imizin decimal değerini toplarsak adresi decimal adresimizin değerini bulabiliriz. Bu noktada programımızı -c komutu ile çalıştırarak sonuçlara ulaşabiliriz ve beklediğimiz gibi olduğunu görebiliriz.

Sonuçlar:

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 4 -c

ARG seed 4
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x0000069a (decimal 1690)
  Limit  : 316

Virtual Address Trace
  VA  0: 0x00000195 (decimal:  405) --> SEGMENTATION VIOLATION
  VA  1: 0x0000009e (decimal:  158) --> VALID: 0x00000738 (decimal: 1848)
  VA  2: 0x00000044 (decimal:   68) --> VALID: 0x000006de (decimal: 1758)
  VA  3: 0x0000019b (decimal:  411) --> SEGMENTATION VIOLATION
  VA  4: 0x000003ab (decimal:  939) --> SEGMENTATION VIOLATION

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 5 -c

ARG seed 5
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00002f79 (decimal 12153)
  Limit  : 415

Virtual Address Trace
  VA  0: 0x0000032e (decimal:  814) --> SEGMENTATION VIOLATION
  VA  1: 0x000003c5 (decimal:  965) --> SEGMENTATION VIOLATION
  VA  2: 0x000002f5 (decimal:  757) --> SEGMENTATION VIOLATION
  VA  3: 0x000003b0 (decimal:  944) --> SEGMENTATION VIOLATION
  VA  4: 0x0000001d (decimal:   29) --> VALID: 0x00002f96 (decimal: 12182)

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 6 -c

ARG seed 6
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x0000349a (decimal 13466)
  Limit  : 459

Virtual Address Trace
  VA  0: 0x000001f0 (decimal:  496) --> SEGMENTATION VIOLATION
  VA  1: 0x0000010b (decimal:  267) --> VALID: 0x000035a5 (decimal: 13733)
  VA  2: 0x00000000 (decimal:    0) --> VALID: 0x0000349a (decimal: 13466)
  VA  3: 0x000002a6 (decimal:  678) --> SEGMENTATION VIOLATION
  VA  4: 0x000001e1 (decimal:  481) --> SEGMENTATION VIOLATION
```

2-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 1 -n 15 -l 868 -c

ARG seed 1
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x00000899 (decimal 2201)
  Limit  : 868

Virtual Address Trace
  VA  0: 0x00000363 (decimal:  867) --> VALID: 0x00000bfc (decimal: 3068)
  VA  1: 0x0000030e (decimal:  782) --> VALID: 0x00000ba7 (decimal: 2983)
  VA  2: 0x00000105 (decimal:  261) --> VALID: 0x0000099e (decimal: 2462)
  VA  3: 0x000001fb (decimal:  507) --> VALID: 0x00000a94 (decimal: 2708)
  VA  4: 0x000001cc (decimal:  460) --> VALID: 0x00000a65 (decimal: 2661)
  VA  5: 0x0000029b (decimal:  667) --> VALID: 0x00000b34 (decimal: 2868)
  VA  6: 0x00000327 (decimal:  807) --> VALID: 0x00000bc0 (decimal: 3008)
  VA  7: 0x00000060 (decimal:   96) --> VALID: 0x000008f9 (decimal: 2297)
  VA  8: 0x0000001d (decimal:   29) --> VALID: 0x000008b6 (decimal: 2230)
  VA  9: 0x00000357 (decimal:  855) --> VALID: 0x00000bf0 (decimal: 3056)
  VA 10: 0x000001bb (decimal:  443) --> VALID: 0x00000a54 (decimal: 2644)
  VA 11: 0x0000030c (decimal:  780) --> VALID: 0x00000ba5 (decimal: 2981)
  VA 12: 0x00000002 (decimal:    2) --> VALID: 0x0000089b (decimal: 2203)
  VA 13: 0x000001c8 (decimal:  456) --> VALID: 0x00000a61 (decimal: 2657)
  VA 14: 0x000002e2 (decimal:  738) --> VALID: 0x00000b7b (decimal: 2939)
```

Görüldüğü gibi en büyük değerimiz VA 0: 867’dir. Eğer limit değerimizi 868 yaparsak tüm değerlerin bound içinde kalması garanti altına almış oluruz.

3-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Relocation$ ./relocation.py -s 0 -n 15 -l 50

ARG seed 0
ARG address space size 1k
ARG phys mem size 16k

Base-and-Bounds register information:

  Base   : 0x0000360b (decimal 13835)
  Limit  : 50

Virtual Address Trace
  VA  0: 0x00000308 (decimal:  776) --> PA or segmentation violation?
  VA  1: 0x000001ae (decimal:  430) --> PA or segmentation violation?
  VA  2: 0x00000109 (decimal:  265) --> PA or segmentation violation?
  VA  3: 0x0000020b (decimal:  523) --> PA or segmentation violation?
  VA  4: 0x0000019e (decimal:  414) --> PA or segmentation violation?
  VA  5: 0x00000322 (decimal:  802) --> PA or segmentation violation?
  VA  6: 0x00000136 (decimal:  310) --> PA or segmentation violation?
  VA  7: 0x000001e8 (decimal:  488) --> PA or segmentation violation?
  VA  8: 0x00000255 (decimal:  597) --> PA or segmentation violation?
  VA  9: 0x000003a1 (decimal:  929) --> PA or segmentation violation?
  VA 10: 0x00000204 (decimal:  516) --> PA or segmentation violation?
  VA 11: 0x00000120 (decimal:  288) --> PA or segmentation violation?
  VA 12: 0x00000305 (decimal:  773) --> PA or segmentation violation?
  VA 13: 0x00000279 (decimal:  633) --> PA or segmentation violation?
  VA 14: 0x00000100 (decimal:  256) --> PA or segmentation violation?

For each virtual address, either write down the physical address it translates to
OR write down that it is an out-of-bounds address (a segmentation violation). For
this problem, you should assume a simple virtual address space of a given size.
```

Pysh mem size: 16K olarak verilmiş 16000 – 50 = 15950

Part 3:

1-)

Burada hesaplamalar için bir java kodu yazdım. Sonuçları program aracılığıyla buldum. Programın kaynak kodlarını da ekledim.

./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 3 için sonuçlar:

30
Binary address: 0011110
Binary address without segment number: 011110
Segment number: 0
Answer is: not valid in Segment 0
69
Binary address: 1000101
Binary address without segment number: 000101
Segment number: 1
Answer is: not valid in Segment 1
47
Binary address: 0101111
Binary address without segment number: 101111
Segment number: 0
Answer is: not valid in Segment 0
77
Binary address: 1001101
Binary address without segment number: 001101
Segment number: 1
Answer is: not valid in Segment 1
80
Binary address: 1010000
Binary address without segment number: 010000
Segment number: 1
Answer is: not valid in Segment 1

segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 4 içi sonuçlar:

30
Binary address: 0011110
Binary address without segment number: 011110
Segment number: 0
Answer is: not valid in Segment 0
13
Binary address: 0001101
Binary address without segment number: 001101
Segment number: 0
Answer is: Valid in Segment 0
50
Binary address: 0110010
Binary address without segment number: 110010
Segment number: 0
Answer is: not valid in Segment 0
19
Binary address: 0010011
Binary address without segment number: 010011
Segment number: 0

Answer is: Valid in Segment 0
8
Binary address: 0001000
Binary address without segment number: 001000
Segment number: 0
Answer is: Valid in Segment 0

segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 5 için sonuçlar:

79
Binary address: 1001111
Binary address without segment number: 001111
Segment number: 1
Answer is: not valid in Segment 1
94
Binary address: 1011110
Binary address without segment number: 011110
Segment number: 1
Answer is: not valid in Segment 1
101
Binary address: 1100101
Binary address without segment number: 100101
Segment number: 1
Answer is: not valid in Segment 1
120
Binary address: 1111000
Binary address without segment number: 111000
Segment number: 1
Answer is: Valid in Segment 1
94
Binary address: 1011110
Binary address without segment number: 011110
Segment number: 1
Answer is: not valid in Segment 1

2-)

En yüksek değer Segment 0 için: 19 (20-1)
En düşük değer Segment 1 için = 108 (128 – 20)
İllegal en düşük değer 20
İllegal en yüksek değer 107

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Segmentation$ ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0 -c -A 19
ARG seed 0
ARG address space size 128
ARG phys mem size 512

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 20

  Segment 1 base  (grows negative) : 0x00000200 (decimal 512)
  Segment 1 limit                  : 20

Virtual Address Trace
  VA  0: 0x00000013 (decimal:   19) --> VALID in SEG0: 0x00000013 (decimal:   19)

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Segmentation$ ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0 -c -A 108
ARG seed 0
ARG address space size 128
ARG phys mem size 512

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 20

  Segment 1 base  (grows negative) : 0x00000200 (decimal 512)
  Segment 1 limit                  : 20

Virtual Address Trace
  VA  0: 0x0000006c (decimal:  108) --> VALID in SEG1: 0x000001ec (decimal:  492)

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Segmentation$ ./segmentation.py -a 128 -p 512 -b 0 -l 20 -B 512 -L 20 -s 0 -c -A 21
ARG seed 0
ARG address space size 128
ARG phys mem size 512

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 20

  Segment 1 base  (grows negative) : 0x00000200 (decimal 512)
  Segment 1 limit                  : 20

Virtual Address Trace
  VA  0: 0x00000015 (decimal:   21) --> SEGMENTATION VIOLATION (SEG0)
```

3-)

./segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 --b0 0 --l0 3 --b1 128 --l1 3 -c

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Segmentation$ ./segmentation.py -a 16 -p 128 -A 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 --b0 0 --l0 3 --b1 128 --l1 3 -c
ARG seed 0
ARG address space size 16
ARG phys mem size 128

Segment register information:

  Segment 0 base  (grows positive) : 0x00000000 (decimal 0)
  Segment 0 limit                  : 3

  Segment 1 base  (grows negative) : 0x00000080 (decimal 128)
  Segment 1 limit                  : 3

Virtual Address Trace
  VA  0: 0x00000000 (decimal:    0) --> VALID in SEG0: 0x00000000 (decimal:    0)
  VA  1: 0x00000001 (decimal:    1) --> VALID in SEG0: 0x00000001 (decimal:    1)
  VA  2: 0x00000002 (decimal:    2) --> VALID in SEG0: 0x00000002 (decimal:    2)
  VA  3: 0x00000003 (decimal:    3) --> SEGMENTATION VIOLATION (SEG0)
  VA  4: 0x00000004 (decimal:    4) --> SEGMENTATION VIOLATION (SEG0)
  VA  5: 0x00000005 (decimal:    5) --> SEGMENTATION VIOLATION (SEG0)
  VA  6: 0x00000006 (decimal:    6) --> SEGMENTATION VIOLATION (SEG0)
  VA  7: 0x00000007 (decimal:    7) --> SEGMENTATION VIOLATION (SEG0)
  VA  8: 0x00000008 (decimal:    8) --> SEGMENTATION VIOLATION (SEG1)
  VA  9: 0x00000009 (decimal:    9) --> SEGMENTATION VIOLATION (SEG1)
  VA 10: 0x0000000a (decimal:   10) --> SEGMENTATION VIOLATION (SEG1)
  VA 11: 0x0000000b (decimal:   11) --> SEGMENTATION VIOLATION (SEG1)
  VA 12: 0x0000000c (decimal:   12) --> SEGMENTATION VIOLATION (SEG1)
  VA 13: 0x0000000d (decimal:   13) --> VALID in SEG1: 0x0000007d (decimal:  125)
  VA 14: 0x0000000e (decimal:   14) --> VALID in SEG1: 0x0000007e (decimal:  126)
  VA 15: 0x0000000f (decimal:   15) --> VALID in SEG1: 0x0000007f (decimal:  127)
```

Part 4:

1-)

alloc() bizim için ayırdığı adresi return edecektir Burada baseAddr değerimiz 1000 olduğu için ilk işlem için 1000 return edilecek. free() ise doğru anlamında 0 return edecektir. İkinci işlem için ise freeList e bakılacak ve 1000 adresine eklenecek yani 1000 return edilecek. Best politikası kullanıldığı boşta olan en küçük blok kullanılacak bu şekilde tahmin yapılarak free list bulunabilir.

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Freespace$ ./malloc.py -n 10 -H 0 -p BEST -s 1 -c
seed 1
size 100
baseAddr 1000
headerSize 0
alignment -1
policy BEST
listOrder ADDRSORT
coalesce False
numOps 10
range 10
percentAlloc 50
allocList
compute True

ptr[0] = Alloc(9)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1009 sz:91 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:91 ]

ptr[1] = Alloc(5)  returned 1000 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1005 sz:4 ] [ addr:1009 sz:91 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:4 ] [ addr:1009 sz:91 ]

ptr[2] = Alloc(1)  returned 1005 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:5 ] [ addr:1006 sz:3 ] [ addr:1009 sz:91 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:1 ] [ addr:1006 sz:3 ] [ addr:1009 sz:91 ]

ptr[3] = Alloc(5)  returned 1000 (searched 4 elements)
Free List [ Size 3 ]:  [ addr:1005 sz:1 ] [ addr:1006 sz:3 ] [ addr:1009 sz:91 ]

Free(ptr[3]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:1 ] [ addr:1006 sz:3 ] [ addr:1009 sz:91 ]

ptr[4] = Alloc(1)  returned 1005 (searched 4 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:5 ] [ addr:1006 sz:3 ] [ addr:1009 sz:91 ]

Free(ptr[4]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:1 ] [ addr:1006 sz:3 ] [ addr:1009 sz:91 ]
```

Görüldüğü gibi küçük boyutlu alloclar geldikçe free listemiz uzamaktadır.

2-)

WORST politikasi en büyük boş bellek bölümünü kullanır. Küçük bloklar geldikçe işlemimiz yavaşlamakta. Daha çok arama yapmak zroundayız.
FIRST ise ilk gördüğü boş belleği kullanır. FIRST kullandığımızda görüldüğü gibi daha hızlı işlem yapılabilmekte çünkü tüm blocklara bakma ihtiyacımız olmamakta.

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Freespace$ ./malloc.py -n 10 -H 0 -p FIRST -s 1 -c
seed 1
size 100
baseAddr 1000
headerSize 0
alignment -1
policy FIRST
listOrder ADDRSORT
coalesce False
numOps 10
range 10
percentAlloc 50
allocList
compute True

ptr[0] = Alloc(9)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1009 sz:91 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:91 ]

ptr[1] = Alloc(5)  returned 1000 (searched 1 elements)
Free List [ Size 2 ]:  [ addr:1005 sz:4 ] [ addr:1009 sz:91 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:4 ] [ addr:1009 sz:91 ]

ptr[2] = Alloc(1)  returned 1000 (searched 1 elements)
Free List [ Size 3 ]:  [ addr:1001 sz:4 ] [ addr:1005 sz:4 ] [ addr:1009 sz:91 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:4 ] [ addr:1005 sz:4 ] [ addr:1009 sz:91 ]

ptr[3] = Alloc(5)  returned 1009 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:4 ] [ addr:1005 sz:4 ] [ addr:1014 sz:86 ]

Free(ptr[3]) returned 0
Free List [ Size 5 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:4 ] [ addr:1005 sz:4 ] [ addr:1009 sz:5 ] [ addr:1014 sz:86 ]

ptr[4] = Alloc(1)  returned 1000 (searched 1 elements)
Free List [ Size 4 ]:  [ addr:1001 sz:4 ] [ addr:1005 sz:4 ] [ addr:1009 sz:5 ] [ addr:1014 sz:86 ]

Free(ptr[4]) returned 0
Free List [ Size 5 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:4 ] [ addr:1005 sz:4 ] [ addr:1009 sz:5 ] [ addr:1014 sz:86 ]

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Freespace$ ./malloc.py -n 10 -H 0 -p WORST -s 1 -c
seed 1
size 100
baseAddr 1000
headerSize 0
alignment -1
policy WORST
listOrder ADDRSORT
coalesce False
numOps 10
range 10
percentAlloc 50
allocList
compute True

ptr[0] = Alloc(9)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1009 sz:91 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:91 ]

ptr[1] = Alloc(5)  returned 1009 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:9 ] [ addr:1014 sz:86 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:5 ] [ addr:1014 sz:86 ]

ptr[2] = Alloc(1)  returned 1014 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:5 ] [ addr:1015 sz:85 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:5 ] [ addr:1014 sz:1 ] [ addr:1015 sz:85 ]

ptr[3] = Alloc(5)  returned 1015 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:5 ] [ addr:1014 sz:1 ] [ addr:1020 sz:80 ]

Free(ptr[3]) returned 0
Free List [ Size 5 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:5 ] [ addr:1014 sz:1 ] [ addr:1015 sz:5 ] [ addr:1020 sz:80 ]

ptr[4] = Alloc(1)  returned 1020 (searched 5 elements)
Free List [ Size 5 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:5 ] [ addr:1014 sz:1 ] [ addr:1015 sz:5 ] [ addr:1021 sz:79 ]

Free(ptr[4]) returned 0
Free List [ Size 6 ]:  [ addr:1000 sz:9 ] [ addr:1009 sz:5 ] [ addr:1014 sz:1 ] [ addr:1015 sz:5 ] [ addr:1020 sz:1 ] [ addr:1021 sz:79 ]
```

3-)

```
toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Freespace$ ./malloc.py -n 6 -A +1,-0,+3,-2,+3,-2 -p BEST -c
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy BEST
listOrder ADDRSORT
coalesce False
numOps 6
range 10
percentAlloc 50
allocList +1,-0,+3,-2,+3,-2
compute True

ptr[0] = Alloc(1)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1001 sz:99 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:99 ]

ptr[1] = Alloc(3)  returned 1001 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1004 sz:96 ]

Invalid Free: Skipping
ptr[2] = Alloc(3)  returned 1004 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1007 sz:93 ]

Free(ptr[2]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:1 ] [ addr:1004 sz:3 ] [ addr:1007 sz:93 ]

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Freespace$ ./malloc.py -n 6 -A +1,-0,+3,-2,+3,-2 -p WORST -c
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy WORST
listOrder ADDRSORT
coalesce False
numOps 6
range 10
percentAlloc 50
allocList +1,-0,+3,-2,+3,-2
compute True

ptr[0] = Alloc(1)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1001 sz:99 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:99 ]

ptr[1] = Alloc(3)  returned 1001 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1004 sz:96 ]

Invalid Free: Skipping
ptr[2] = Alloc(3)  returned 1004 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1007 sz:93 ]

Free(ptr[2]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:1 ] [ addr:1004 sz:3 ] [ addr:1007 sz:93 ]

toor@001:~/Desktop/461 - Isletim Sistemleri/Odev3/HW-Freespace$ ./malloc.py -n 6 -A +1,-0,+3,-2,+3,-2 -p FIRST -c
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy FIRST
listOrder ADDRSORT
coalesce False
numOps 6
range 10
percentAlloc 50
allocList +1,-0,+3,-2,+3,-2
compute True

ptr[0] = Alloc(1)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1001 sz:99 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:99 ]

ptr[1] = Alloc(3)  returned 1001 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1004 sz:96 ]

Invalid Free: Skipping
ptr[2] = Alloc(3)  returned 1004 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1007 sz:93 ]

Free(ptr[2]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:1 ] [ addr:1004 sz:3 ] [ addr:1007 sz:93 ]
```

./malloc.py -n 6 -A +1,-0,+3,-2,+3,-2 -c denedğimizde tüm policylerde aynı şekilde Free List 3 uzunluğunda olmaktadır.