دانشکده‌ی مهندسی کامپیوتر

سیستم‌های عامل
تمرین‌های سری پنجم کارگاه

علی حیدری، محمدجواد میرشکاری

۱۶ خرداد ۱۳۹۸

۱

First run with the flags `-n 10 -H 0 -p BEST -s 0` to generate a few random allocations and frees. Can you predict what alloc()/ free() will return? Can you guess the state of the free list after each request? What do you notice about the free list over time?

پاسخ. بله می‌توان پیش‌بینی کرد. در مرحله‌ی اول سه بایت اختصاص می‌دهیم. پس فضای ما از ۱۰۰۳ به اندازه‌ی ۹۷ خالی خواهد بود. در مرحله‌ی بعد فضای اختصاص داده شده راخالی می‌کنیم. حافظه‌ی خالی به دو قسمت خواهد بود. یکی از ۱۰۰۰ به اندازه‌ی ۳ و یکی از ۱۰۰۳ به اندازه‌ی ۲ زیرا از Coalesca استفاده نکردیم. بقیه‌ی مراحل هم به همین ترتیب خواهد بود. باید توجه کنیم که سیاست ما برای تخصیص فضای خالی Best fit است. و هنگام تخصیص فضا کوچک‌ترین قسمتی که به اندازه‌ی کافی بزرگ باشد تخصیص داده خواهد شد. پس باتوجه به این می‌توان پیش‌بینی کرد که چه فضایی هنگام درخواست تخصیص داده می‌شود.
هر چه جلوتر برویم تعداد عناصر free list بیش‌تر می‌شود و حافظه‌ی خالی به قسمت‌های بیش‌تری تقسیم می‌شود که دلیل آن مربوط به عدم استفاده از Coalesca می‌باشد.

```
ali@DESKTOP:~$ python2.7 malloc.py -n 10 -H 0 -p BEST -s 0 -c
seed 0
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(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:97 ]

ptr[1] = Alloc(5)  returned 1003 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1008 sz:92 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:92 ]

ptr[2] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:84 ]

ptr[3] = Alloc(8)  returned 1008 (searched 4 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[3]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:84 ]

ptr[4] = Alloc(2)  returned 1000 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:84 ]

ptr[5] = Alloc(7)  returned 1008 (searched 4 elements)
```

```
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1015 sz:1 ] [
    addr:1016 sz:84 ]
```

۲

How are the results different when using a WORST fit policy to search the free list (`-p WORST`)? What changes?

پاسخ. به جای این که کوچک‌ترین قسمت حافظه را اختصاص بدهیم بزرگ‌ترین قسمت حافظه ر انتخاب می‌کنیم بنابراین سرعت رشد عناصر free list بیش‌تر می‌شود.

```
ali@DESKTOP:~$ python2.7 malloc.py -n 10 -H 0 -p WORST -s 0 -c
seed 0
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(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:97 ]

ptr[1] = Alloc(5)  returned 1003 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1008 sz:92 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:92 ]

ptr[2] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:84 ]

ptr[3] = Alloc(8)  returned 1016 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1024 sz:76 ]

Free(ptr[3]) returned 0
Free List [ Size 5 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:8 ] [ addr:1024 sz:76 ]

ptr[4] = Alloc(2)  returned 1024 (searched 5 elements)
Free List [ Size 5 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:8 ] [ addr:1026 sz:74 ]
```

```
ptr[5] = Alloc(7)  returned 1026 (searched 5 elements)
Free List [ Size 5 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:8 ] [ addr:1033 sz:67 ]
```

۳

What about when using FIRST fit (`-p FIRST`)? What speeds up when you use first fit?

پاسخ. از آن‌جایی که هنگام استفاده از first fit لازم نیست همه‌ی free list را پیمایش کنیم پس سرعت تخصیص دادن حافظه افزایش پیدا خواهد کرد.

```
ali@DESKTOP:~$ python2.7 malloc.py -n 10 -H 0 -p FIRST -s 0 -c
seed 0
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(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:97 ]

ptr[1] = Alloc(5)  returned 1003 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:3 ] [ addr:1008 sz:92 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:92 ]

ptr[2] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:84 ]

ptr[3] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1016 sz:84 ]

Free(ptr[3]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:3 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:84 ]

ptr[4] = Alloc(2)  returned 1000 (searched 1 elements)
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1008 sz:8 ] [
    addr:1016 sz:84 ]
```

```
ptr[5] = Alloc(7)  returned 1008 (searched 3 elements)
Free List [ Size 4 ]:  [ addr:1002 sz:1 ] [ addr:1003 sz:5 ] [ addr:1015 sz:1 ] [
    addr:1016 sz:84 ]
```

۴

For the above questions, how the list is kept ordered can affect the time it takes to find a free location for some of the policies. Use the different free list orderings (`-l ADDRSORT`, `-l SIZESORT+`, `-l SIZESORT-`) to see how the policies and the list orderings interact.

پاسخ. پس از اجرای دستورهای قبلی با ADDRSORT orderings زمان اجرا برای هر دو Worth و Best تغییر چندانی نکرد. اما برای first fit زمان اجرا کمی کاهش می‌یابد به این دلیل که اگر بزرگ‌ترین خانه اول باشد پیدا کردن خانه خالی با فضای مناسب راحت‌تر انجام می‌پذیرد worth fit با مرتب‌سازی SIZESORT هم نیازی به پیمایش تمام free list ندارد.

```
ali@DESKTOP:~$ python2.7 malloc.py -n 10 -H 0 -p BEST -s 0 -c -l ADDRSORT,-l
    SIZESORT+, -l SIZESORT-
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy BEST
listOrder SIZESORT-
coalesce False
numOps 10
range 10
percentAlloc 50
allocList
compute True

ptr[0] = Alloc(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1003 sz:97 ] [ addr:1000 sz:3 ]

ptr[1] = Alloc(5)  returned 1003 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1008 sz:92 ] [ addr:1000 sz:3 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1008 sz:92 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[2] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1016 sz:84 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1016 sz:84 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ]
    [ addr:1000 sz:3 ]

ptr[3] = Alloc(8)  returned 1008 (searched 4 elements)
Free List [ Size 3 ]:  [ addr:1016 sz:84 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

Free(ptr[3]) returned 0
Free List [ Size 4 ]:  [ addr:1016 sz:84 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ]
    [ addr:1000 sz:3 ]
```

```
ptr[4] = Alloc(2)  returned 1000 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1016 sz:84 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ]
    [ addr:1002 sz:1 ]

ptr[5] = Alloc(7)  returned 1008 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1016 sz:84 ] [ addr:1015 sz:1 ] [ addr:1003 sz:5 ]
    [ addr:1002 sz:1 ]
```

```
ali@DESKTOP:~$ python2.7 malloc.py -n 10 -H 0 -p WORST -s 0 -c -l ADDRSORT, -l
    SIZESORT+, -l SIZESORT-
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy WORST
listOrder SIZESORT-
coalesce False
numOps 10
range 10
percentAlloc 50
allocList
compute True

ptr[0] = Alloc(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1003 sz:97 ] [ addr:1000 sz:3 ]

ptr[1] = Alloc(5)  returned 1003 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1008 sz:92 ] [ addr:1000 sz:3 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1008 sz:92 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[2] = Alloc(8)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1016 sz:84 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1016 sz:84 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ]
    [ addr:1000 sz:3 ]

ptr[3] = Alloc(8)  returned 1016 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1024 sz:76 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ]
    [ addr:1000 sz:3 ]

Free(ptr[3]) returned 0
Free List [ Size 5 ]:  [ addr:1024 sz:76 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ]
    [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[4] = Alloc(2)  returned 1024 (searched 5 elements)
Free List [ Size 5 ]:  [ addr:1026 sz:74 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ]
    [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[5] = Alloc(7)  returned 1026 (searched 5 elements)
Free List [ Size 5 ]:  [ addr:1033 sz:67 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ]
    [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]
```

```
ali@DESKTOP:~$ python2.7 malloc.py -n 10 -H 0 -p FIRST -s 0 -c -l ADDRSORT, -l
```

```
    SIZESORT+, -l SIZESORT-
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy FIRST
listOrder SIZESORT-
coalesce False
numOps 10
range 10
percentAlloc 50
allocList
compute True

ptr[0] = Alloc(3)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1003 sz:97 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1003 sz:97 ] [ addr:1000 sz:3 ]

ptr[1] = Alloc(5)  returned 1003 (searched 1 elements)
Free List [ Size 2 ]:  [ addr:1008 sz:92 ] [ addr:1000 sz:3 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1008 sz:92 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[2] = Alloc(8)  returned 1008 (searched 1 elements)
Free List [ Size 3 ]:  [ addr:1016 sz:84 ] [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1016 sz:84 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ]
    [ addr:1000 sz:3 ]

ptr[3] = Alloc(8)  returned 1016 (searched 1 elements)
Free List [ Size 4 ]:  [ addr:1024 sz:76 ] [ addr:1008 sz:8 ] [ addr:1003 sz:5 ]
    [ addr:1000 sz:3 ]

Free(ptr[3]) returned 0
Free List [ Size 5 ]:  [ addr:1024 sz:76 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ]
    [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[4] = Alloc(2)  returned 1024 (searched 1 elements)
Free List [ Size 5 ]:  [ addr:1026 sz:74 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ]
    [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]

ptr[5] = Alloc(7)  returned 1026 (searched 1 elements)
Free List [ Size 5 ]:  [ addr:1033 sz:67 ] [ addr:1008 sz:8 ] [ addr:1016 sz:8 ]
    [ addr:1003 sz:5 ] [ addr:1000 sz:3 ]
```

۵

Coalescing of a free list can be quite important. Increase the number of random allocations (say to `-n 1000`). What happens to larger allocation requests over time? Run with and without coalescing (i.e., without and with the `-C` flag). What differences in outcome do you see? How big is the free list over time in each case? Does the ordering of the list matter in this case?

پاسخ. به دلیل این که بدون Coalescing، free list ما با قسمت‌های کوچک پر شده حافظه پر می‌شود، درخواست‌های بزرگ برای گرفتن حافظه fail می‌شوند.

۶

What happens when you change the percent allocated fraction `-P` to higher than 50? What happens to allocations as it nears 100? What about as the percent nears 0?

پاسخ. اندازه‌ی free list هنگام نزدیک شدن P به ۱۰۰ به ۱ میل می‌کند و هنگام نزدیک شدن P به صفر زیاد می‌شود. درواقع P نسبت دستورات درخواست‌کننده‌ی حافظه را مشخص می‌کند.

```
ali@DESKTOP:~$ python2.7 malloc.py n 10 -H 0 -p BEST -s 0 -c -P 5
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy BEST
listOrder ADDRSORT
coalesce False
numOps 10
range 10
percentAlloc 5
allocList
compute True

ptr[0] = Alloc(8)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1008 sz:92 ]

Free(ptr[0]) returned 0
Free List [ Size 2 ]:  [ addr:1000 sz:8 ] [ addr:1008 sz:92 ]

ptr[1] = Alloc(5)  returned 1000 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1005 sz:3 ] [ addr:1008 sz:92 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:3 ] [ addr:1008 sz:92 ]

ptr[2] = Alloc(6)  returned 1008 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:3 ] [ addr:1014 sz:86 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:3 ] [ addr:1008 sz:6 ] [
    addr:1014 sz:86 ]

ptr[3] = Alloc(7)  returned 1014 (searched 4 elements)
Free List [ Size 4 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:3 ] [ addr:1008 sz:6 ] [
    addr:1021 sz:79 ]

Free(ptr[3]) returned 0
Free List [ Size 5 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:3 ] [ addr:1008 sz:6 ] [
    addr:1014 sz:7 ] [ addr:1021 sz:79 ]

ptr[4] = Alloc(1)  returned 1005 (searched 5 elements)
Free List [ Size 5 ]:  [ addr:1000 sz:5 ] [ addr:1006 sz:2 ] [ addr:1008 sz:6 ] [
    addr:1014 sz:7 ] [ addr:1021 sz:79 ]
```

```
Free(ptr[4]) returned 0
Free List [ Size 6 ]:  [ addr:1000 sz:5 ] [ addr:1005 sz:1 ] [ addr:1006 sz:2 ] [
    addr:1008 sz:6 ] [ addr:1014 sz:7 ] [ addr:1021 sz:79 ]
```

```
ali@DESKTOP:~$ python2.7 malloc.py n 10 -H 0 -p BEST -s 0 -c -P 95
seed 0
size 100
baseAddr 1000
headerSize 0
alignment -1
policy BEST
listOrder ADDRSORT
coalesce False
numOps 10
range 10
percentAlloc 95
allocList
compute True

ptr[0] = Alloc(8)  returned 1000 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1008 sz:92 ]

ptr[1] = Alloc(3)  returned 1008 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1011 sz:89 ]

ptr[2] = Alloc(5)  returned 1011 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1016 sz:84 ]

ptr[3] = Alloc(4)  returned 1016 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1020 sz:80 ]

ptr[4] = Alloc(6)  returned 1020 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1026 sz:74 ]

ptr[5] = Alloc(6)  returned 1026 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1032 sz:68 ]

ptr[6] = Alloc(8)  returned 1032 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1040 sz:60 ]

ptr[7] = Alloc(3)  returned 1040 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1043 sz:57 ]

ptr[8] = Alloc(10)  returned 1043 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1053 sz:47 ]

ptr[9] = Alloc(10)  returned 1053 (searched 1 elements)
Free List [ Size 1 ]:  [ addr:1063 sz:37 ]
```

۷

What kind of specific requests can you make to generate a highly fragmented free space? Use the `-A` flag to create fragmented free lists, and see how different policies and options change the organization of the free list.

پاسخ.

```
ali@DESKTOP:~$ python2.7 malloc.py -n 6 -A +1,-0,+2,-1,+3,-2 -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,+2,-1,+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(2)  returned 1001 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1003 sz:97 ]

Free(ptr[1]) returned 0
Free List [ Size 3 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:2 ] [ addr:1003 sz:97 ]

ptr[2] = Alloc(3)  returned 1003 (searched 3 elements)
Free List [ Size 3 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:2 ] [ addr:1006 sz:94 ]

Free(ptr[2]) returned 0
Free List [ Size 4 ]:  [ addr:1000 sz:1 ] [ addr:1001 sz:2 ] [ addr:1003 sz:3 ] [
    addr:1006 sz:94 ]
```