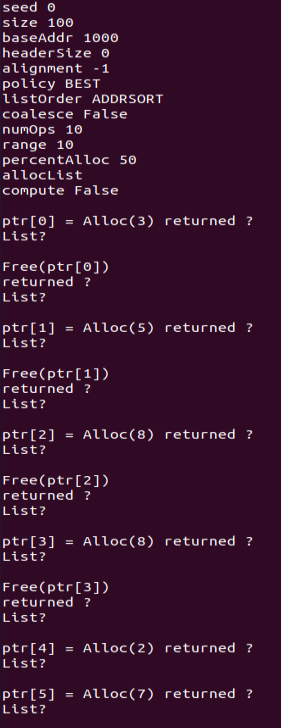
**Homework Wan Huzaifah bin Wan Azhar**

**Answer:**



Ptr[0] = Alloc(3) returned 1000  
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  
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  
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  
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  
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  
Free List [Size 4]: [addr: 1002 sz: 1] [addr: 1003 sz: 5] [addr: 1015 sz: 1] [addr: 1016 sz: 84]



* The size of free list continually increasing as the memory allocator allocate memory free largest free list
* The memory allocator rarely reused the memory that is freed by previous program.



* FIRST fit works like BEST fit. However, it is faster as it checks the first available fit rather than the most optimal fit.
* It can be seen from allocating ptr 3, 4 and 5 that it searches faster. Best fit search size of 4 while Fast fit search size of 3, 1, 3



* ADDRSORT sort the list of size according to address (which is default)
* SIZESORT+ sort the list of size by lowest to highest while SIZESORT- sort from highest to lowest.
* The order of list is important as it can speed up the search. For example, if worst fit is chosen and the list is sorted by SIZESORT-, each allocation will take about 1 time as the highest is the first in the list.



* Without coalescing, the size of the free list will continually increasing as the free blocks will not merge into bigger blocks.
* Without using the -C parameter, the size of the free list goes up to 34. But, with -C parameter, final size of free list is 4.
* Order of the list matter because the address cannot merge address that is not near them. As such, ordering the list by size for example, will not coalesce the list.



* Percent allocated fraction of 100 allocate all the pointer but never frees it.
* This means that it allocates all of the data is in the memory.
* Percent allocated fraction 1 allocates the pointer but frees it quickly.



* Python malloc.py -S 50 -n 10 -H 0 -p BEST -A +10,-0,+3,+8,-1,+9,+10,-2,+11 -c