2.2bash-awk-lab

CST 334 (Operating Systems)
Dr. Glenn Bruns

Lab: AWK

1. Copy this file to your home directory on hosting:

```
/home/CLASSES/brunsglenn/cst334/labs/awk-lab/malloc-out.txt
```

2. The file is output from an OSTEP memory allocator simulator. Look at the first lines of the file:

```
$ head malloc-out.txt
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(7)  returned 1001 (searched 2 elements)
Free List [ Size 2 ]:  [ addr:1000 sz:1 ] [ addr:1008 sz:92 ]

Free(ptr[1]) returned 0
```

The memory allocator maintains a "free list" of available chunks of memory. A request for memory has the form `Alloc(n)`, where n is the number of requested bytes. The first couple of lines show a request for 1 byte of memory. A pointer to that byte is returned, and then the free list contains a single element: a chunk of 99 bytes, located at address 1001.

A release of memory back to the allocator has the form `Free(ptr)`, where ptr is the address of the chunk of memory to be returned. In lines 4-5 above, after the chunk (which happens to be only 1 byte) is returned, the memory allocator now has two elements in its free list.

3. Now write awk scripts to process malloc-out.txt:

   ○ write an awk script `freesize.awk` that gets the size of the free list after every “Free” or “Alloc” operation. Your program should act like this:

```
$ awk -f freesize.awk malloc-out.txt | wc -l
1000
$ awk -f freesize.awk malloc-out.txt | tail -5
41
42
41
42
43
```

○ write an awk script `count_allocs.awk` that counts the number of successful allocs and the number of failed alloc calls. Your program should act like this:

```
$ awk -f count_allocs.awk malloc-out.txt
num successes: 448; num failures: 106
```

○ write an awk script `num_bytes.awk` that records the number of bytes requested in each alloc call. Your program should act like this:

```
$ awk -f num_bytes.awk malloc-out.txt | wc -l
554
$ awk -f num_bytes.awk malloc-out.txt | head -200 | tail -5
1
10
3
10
6
```

○ write an awk script `succ_reqs.awk` that prints the number of bytes requested, and then a 1 or a 0 depending on whether the request was successful (1 means success). Your program should act like this:

```
$ awk -f succ_reqs.awk malloc-out.txt | tail -5
8 0
3 1
5 1
10 0
6 1
```

○ write an awk script `list_sizes.awk` that prints the size of every element in the free list, in order, after each Free or Alloc operation. Your program should act like this:

```
$ awk -f list_sizes.awk malloc-out.txt | head

99
1 99
1 92
1 7 92
1 2 92
1 2 84
1 2 8 84
1 5 2 8 84
5 2 8 84
2 8 84
```

Hint: you can use 'printf' and loops in awk programs. In both cases the syntax is similar to C. Here is an example of an awk program with a loop:

```
{ n = $1
  for (i=0; i < n; i++) {
     printf("%s ", $(2+i))
  }
```

```
}
```

This program assigns the value in field 1 to variable n, then prints fields 2, 3, 4 up to field 2 + n - 1.