CS117 (Fall 2017) Programming Assignment 5 30 pts. Due: Oct. 30 (M) – Lab1
Nov.02(Th) – Lab2

Simulation for dynamic memory (heap) allocation/deallocation (garbage collection).

In the memory allocation part, linked-lists are manipulated. For the memory deallocation, we use the lazy garbage collection mechanism that has **mark** and **sweep** phases.

Assume that the size of the simulated dynamic memory is 10 cells and each cell consists of three fields, i.e. key, next, and mark_bit (initialized with 0).

For this practice, we use two linked lists (list1, list2) whose head pointers are named L1 and L2. Initially, the free-list (head pointer name: Free) contains all the cells and L1 = -1, L2 = -1, and Free = 1. The initial memory configuration is:

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
key										
next	2	3	4	5	6	7	8	9	10	-1
mark	0	0	0	0	0	0	0	0	0	0

L1 = -1, L2 = -1, Free = 1 //head pointers for list1, list2, and free-list

Part1: memory allocation

After processing the following insertion (attach) operations consecutively,

```
insert (L1, 3);  //insert (attach) a node with key value 3 into list1
insert (L1, 1);  //insert (attach) a node with key value 1 into list1
insert (L2, 4);
insert (L1, 5);
insert (L2, 2);
insert (L2, 9);
insert (L2, 8);
insert (L1, 4)
```

the resulting memory configuration and head pointers are:

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
key	3	1	4	5	2	9	8	4		
next	2	4	5	8	6	7	-1	-1	10	-1
mark	0	0	0	0	0	0	0	0	0	0

L1 = 1, L2 = 3, Free = 9

- Write a menu-driven program for the following menu options:
  Print_memory //displays memory contents and values of pointers (L1, L2, Free)
  Insert (Head_pointer, key) //gets a new node from the free-list and attaches it to the list

Test your program using the above 8 insertion operations (keep the order). Show the memory contents and pointer (L1, L2, Free) values after each operation, by invoking the print-memory option.

Part2: memory deallocation

Include the following two additional options into the program developed in part1:

Delete (Head_pointer, key) //deletes the node with key from the list pointed to by Head_pointer
Garbage_Collect ( ) //mark-and-sweep garbage collection

For the garbage-collection, use the ***mark-and-sweep*** mechanism, i.e.,

Mark phase: Trace all reachable nodes starting from all head pointers (L1, L2, Free – in our practice), and mark all reachable nodes.

Sweep phase: Starting from the lowest memory address (memory[1]), collect all unmarked nodes and return them to the free-list (to the head of the free-list each time).

Operations on the free-list are LIFO (last in first out, like stack), i.e., garbage collector collects a garbage node and places it to the head of the free-list (push like). When a memory allocation is requested, the 1st free node is assigned (*pop like*).

Test your program (part2) using the following sequence of operations:

delete (L1, 4); print_memory;
delete (L2, 8); print_memory;
delete (L1, 1); print_memory;
delete (L2, 4); print_memory;
delete (L1, 5); print_memory;
garbage-collect( );
print_memory

After performing the above five delete operations (right before the garbage collection), the memory configuration is:

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
key	3	1	4	5	2	9	8	4		
next	-1	4	5	-1	6	-1	-1	-1	10	-1
mark	0	0	0	0	0	0	0	0	0	0

L1 = 1, L2 = 5, Free = 9 //head pointers for list1, list2, and free-list

Memory configuration after the garbage collection is not shown in this assignment sheet. Please make it by yourself.

- **Write a menu-driven program (make one program which includes both part1 and part2), and run your program with the given data (8 insertions and 5 deletions) shown in this sheet.**
- **Include good documentation (global and each function head) and submit a zip file containing the source code file and run time output by Email to:** jpark@csufresno.edu

  Please make your zip file name and the email subject field as the following:
  CS117-Prog5-yourFirstName-yourLastName.zip