Name: *(as it would appear on official course roster)*		
Umail address:	@umail.ucsb.edu	section
Optional: name you wish to be called if different from name above.		
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"		

1

h08: Chapter 13 Linked lists

h08

CS16 S18

ready?	assigned	due	points
true	Fri 05/18 09:00AM	Fri 05/25 11:59PM	52

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE, OR IF APPLICABLE, SUBMITTED ON GRADESCOPE. There is NO MAKEUP for missed assignments; in place of that, we drop the three lowest scores (if you have zeros, those are the three lowest scores.)

Read Chapter 13, section 13.1 (pages 740- 759). You don't need to turn this homework in. To earn credit for this homework, complete the corresponding quiz on gauchospace AFTER you have completed the pen and pencil version of the homework. The quiz will be available one day before the due date indicated on the homework.

PLEASE MARK YOUR HOMEWORK CLEARLY, REGARDLESS OF IF YOU WRITE IT OUT IN INK OR PENCIL!
FOR BEST RESULTS, PRINT THIS PAGE AS A PDF, THEN PRINT THE PDF

Please:

- **No Staples**.
- **No Paperclips**.
- **No folded down corners**.

8.(2 pts) Consider the 'head' variable on page 741. What is the value of head for an empty list? null

4.(6 pts) Assume you are given a pointer to the head of an existing list (named head). The nodes of the linked-list are of type struct Node (as defined on display 13.7 on page 754). Write a for-loop to iterate through the list and print the data of every other element of the list (starting with the first element).

```
for ( Node *t = head; t != 0 && t->next != 0;
      t = t->next->next )
    cout << t->data;
```

5.(12 pts) Consider a linked list where each node is of the same type as in the previous question. Complete the definition of the function deleteNode given below, that takes as input a pointer to the head of the list, and an integer value. The function should delete all the nodes in the list whose data members have the given value. If the list is empty or if there is no node with the given value, the function should not do anything.

```
void deleteNode(struct Node*& head, int value);
```

```
if (head == null) return
Node *this_node = head;
deleteNode (head->next, value);
// delete the value from the rest of the list
// the next pointer is automatically updated
```

// because it is passed by reference

```
if (head->data == value) {
    Node * newhead = head->next;
    delete head;
    head = newhead;
}
```

For all the following questions, use the definitions of the struct Node and struct LinkedList from lab 06.

6.(10 pts) Implement a function that returns the number of even elements in a linked list. Test your code before writing it out. Illegible code will receive 0 credit. Below is the declaration of the function

```
int countEven(LinkedList* list);
```

7.(2 pts) In the implementation of a linked list, why does struct Node contain a pointer member variable of type Node *?

Because it points to the next node in the linked list by storing the address of the next Node

Q6

```
int countEven(LinkedList *list){
    int count = 0;
    Node *h = list->head;
    while(h){
        if(h->data % 2 == 0)
            count++
        h = h->next;
    }
}
```

2

h08

CS16 S18

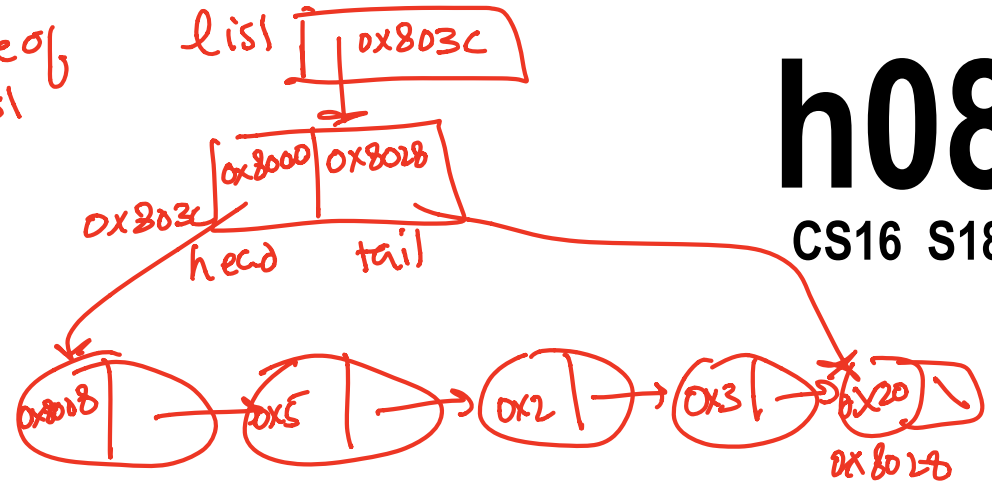
3

h08

CS16 S18

8.(10 pts) You are given the following memory map, where the numbers on the left column are memory locations and the numbers on the right are 4 byte values stored at each location. Suppose that a linked-list is stored in the given memory segment. In the memory map there exists a pointer (named 'list') to a LinkedList object. You are given the location of list to be 0x8008. Draw the pointer diagram for the linked list showing all the nodes in the linked-list, the data stored in each node and all pointers, starting with the variable 'list'. The LinkedList and Node types are the same as those defined in lab06

Memory Address	Value
0x8000	0x8008
0x8004	0x8020
0x8008	0x803C
0x800C	0x000A
0x8010	0x8018
0x8014	0x0002
0x8018	0x8030
0x801C	0x0004
0x8020	0x0005
0x8024	0x8014
0x8028	0x0020
0x802C	0x0000
0x8030	0x0003
0x8034	0x8028
0x8038	0x8008
0x803C	0x8000
0x8040	0x8028



9.(10 pts) Implement a function that takes a linked list and the number of elements in the list as input and returns a dynamic array containing the data elements of the linked list. Test your code before writing it out.

```
int* linkedListToArray(LinkedList * list, int len);
```



```
if (len <= 0)
    return null;
int i=0;
int *arr = new int[len];
Node * h = list->head;
while (h){
    arr[i] = h->data;
    h = h->next;
    i++;
}
}
```