University of Toronto
csc148S—Introduction to Computer Science, Spring 1997

First Term Test — Horton's section

Sample Solutions!

Duration: 50 minutes

Aids allowed: None

Make sure that your examination booklet has 5 pages (including this one). Write your answers in the spaces provided. Write legibly.

Family Name: ______________________ **Lecturer:** ______________________

First Name: ______________________ **Tutor:** ______________________

Student # : ______________________

1.	________	/ 13
2.	________	/ 7
3.	________	/ 6
Total	________	/ 26

Question 1

[13 marks in total]

Consider the following type and variable, useful for creating a linked list of integers:

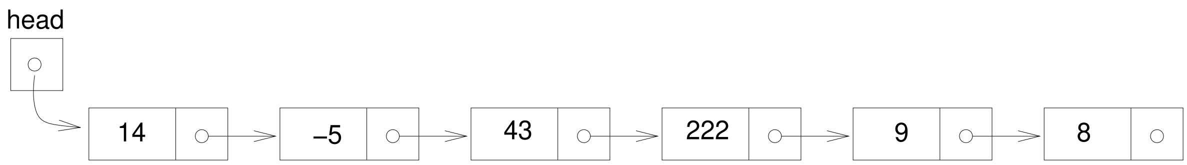
```
typedef struct nodetyp {
    int             data;             /* The integer for this node */
    struct nodetyp *next;             /* Pointer to the next node in list */
    } Node;
Node *head;
```

The following code operates on a linked list pointed to by `head`:

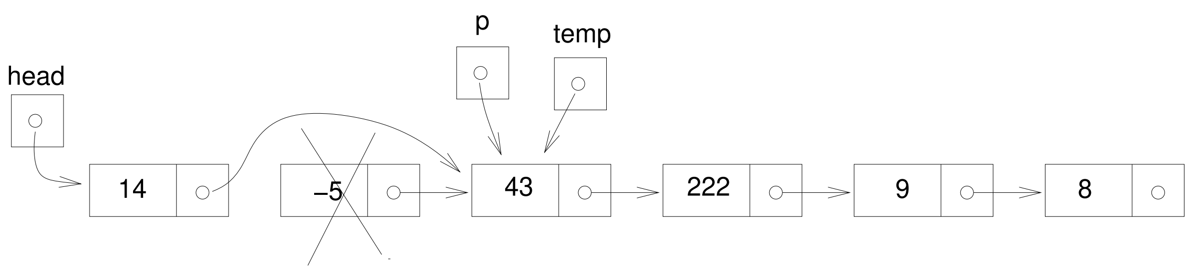
```
Node *p = head;
while (p != NULL)
{
   Node *temp = p->next;
   if (temp != NULL) {
        temp = temp->next;
   }
   free(p->next);    /* Free one node */
   p->next = temp;
   p = p->next;
}
```

(a) [6 marks]

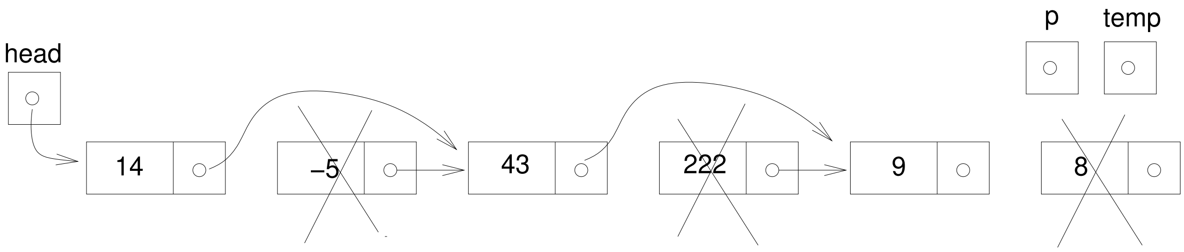
Suppose that `head` points to the following linked list:



and that the code above is then executed. Draw a picture below of the linked list and the variables `head`, `p` and `temp` after the first iteration of the `while` loop has been completed.



Draw a picture below of the linked list and the variables `head`, `p` and `temp` after the code has been completely executed.



(b) [3 marks]
Write an appropriate comment to go above this block of code.

`Removes and frees every second element of the linked list pointed to by 'head', starting with the second element.`

(c) [3 marks]
What test cases would you use for testing this block of code? Briefly explain the purpose of each test case.

`The content of the nodes does not affect the operation of this code at all, so it's not necessary to vary the content. However, the length of the linked list must be varied. Lists with even and odd size should be tried, since the code does something to every second element.`

- `list of length 0`
  `(empty list -- important border case)`
- `lists of length 1, 2, 3, and 4`
  `(other small lists of even and odd size -- border cases)`
- `larger list with even size`
- `larger list with odd size`

(d) [1 mark]
Declare a node type for a doubly-linked list of integers.

```
typedef struct nodetyp {
    int             data;
    struct nodetyp *forward, *backward;
    } Node;
```

Question 2

[7 marks]

Using the `Node` type defined in question 1, write a C function that returns the sum of the values in a linked list of integers. Your function should return 0 if the linked list is empty.

```
int sumUp (Node *head)
{
     Node *p = head;
     int sum = 0;

     while (p)
     {
          sum += p->data;
          p = p->next;
     }

     return sum;
}
```

Question 3

[6 marks in total]

When implementing an ADT (such as an Ordered Set) in a C program, one must decide where to put the data structures and functions. In class we talked about two general approaches: (1) define the data structure variables in the .c file, or (2) just declare a type, and put that type in the .h file.

(a) [2 marks]
With method (2), where are actual instances of the ADT defined?

`The client code defines the instances itself, outside the ''module'' (thus they are not defined in either the .c or the .h file).`

(b) [4 marks]
What is the main tradeoff between methods (1) and (2)?

`With method (1) we get very good information-hiding, but we can have only one instance of the ADT.`

`With method (2) we can have multiple instances, but we lose the information-hiding. Not only does the client have to see and use the type (to pass instances of the ADT into the module's functions), but nothing prevents the client from accessing the data structure directly rather than by using a function from the module.`

Your answers will probably fit above this line.