

ENCM 339 Fall 2005, T02 (Dr. Norman's tutorial) Tues., Nov. 22, Exercise 2 Solution

This is the solution I promised to post on the Web. The point of this exercise is to help you understand assignments involving pointers when a linked list is constructed—if you're struggling trying to figure where all the arrows point, it may be helpful to pick some numbers to use as addresses.

Below is the program. I added `point two` so could show more steps in the evolution of the linked list:

```
#include <stdlib.h>
struct node { int item; struct node *next; };

struct node * insert(struct node *head, int new_item)
{
    struct node *p;
    for (p = head; p != NULL && p->item != new_item; p = p->next)
        ;

    /* point one */

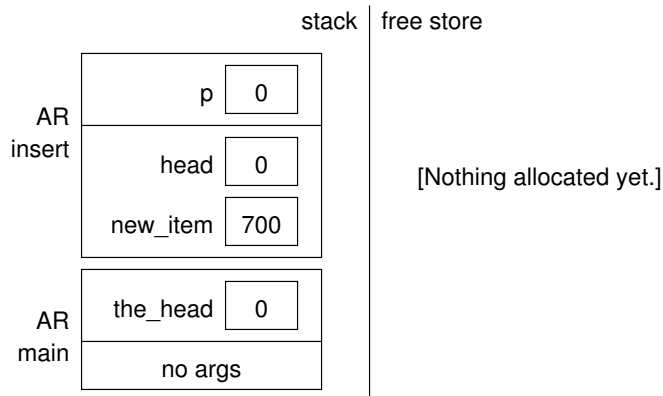
    if (p == NULL) {
        p = (struct node *) malloc(sizeof(struct node));
        p->next = head;
        p->item = new_item;
        head = p;
    }

    /* point two */

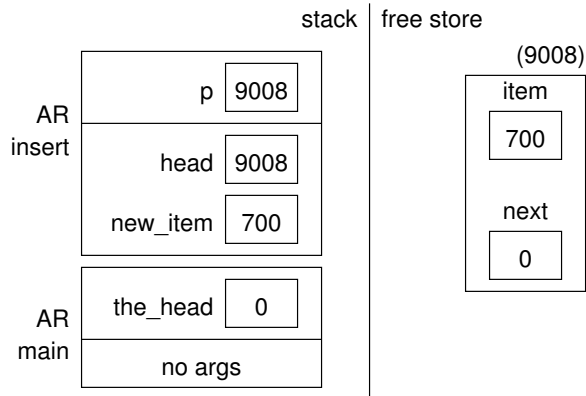
    return head;
}

int main(void)
{
    struct node *the_head = NULL;
    the_head = insert(the_head, 700);
    the_head = insert(the_head, 900);
    the_head = insert(the_head, 800);
    the_head = insert(the_head, 700);
    return 0;
}
```

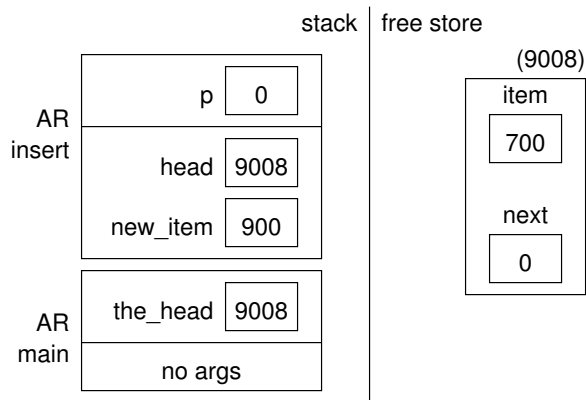
Here is the **first time** the program gets to **point one**. Note that `p == NULL` is true, so `insert` is about to allocate a node in the `if` statement:



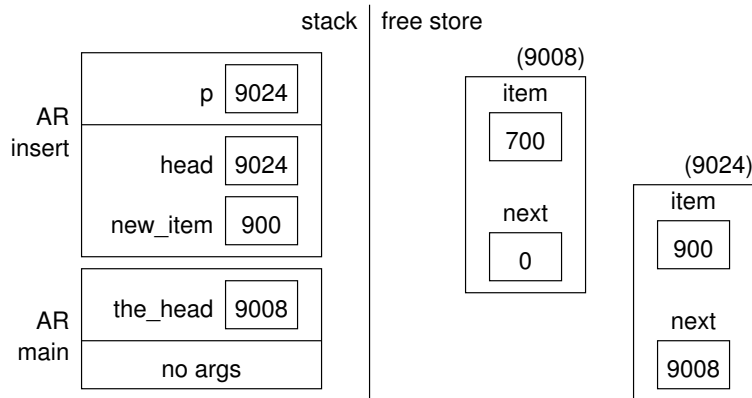
Here is the **first time** the program gets to **point two**. Note that a one-node linked list has been created, and the address of its head node is about to be returned to **main**:



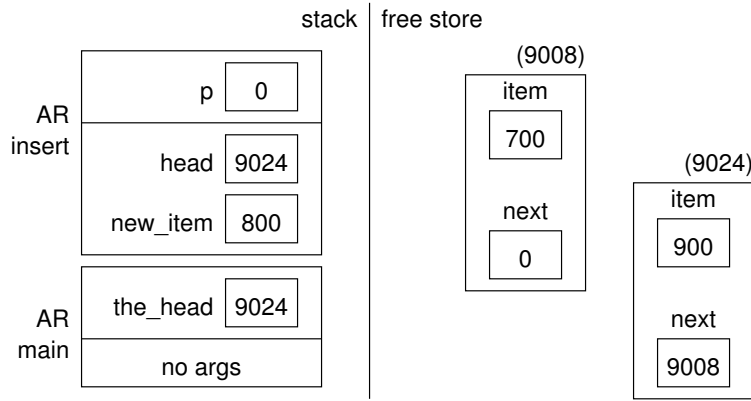
Point one, second time.



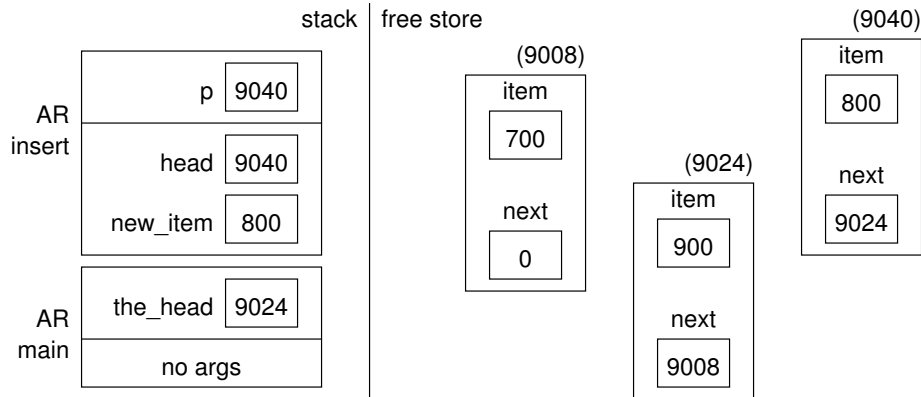
Point two, second time. Note that a new node with an item value of 900 has been inserted in front of the original head node.



Point one, third time.



Point two, third time.



Point one, last time. This time there will be no allocation or insertion—the expression `p == NULL` is false. Essentially, `insert` inserts a new node at the head of a list, but only if the new item doesn't match any items that are already in the list.

