C - EXERCISE MODULE 6

Cagara, Johann Miguel
Gabinete, Keith Ginoel

Authors' Note:

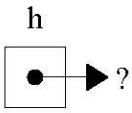
In line with a particular instruction in submitting C programs for the collaborative exercises in CMSC 21 wherein each of the two partners must indicate in the documentation part of each program the particular contributions one has done for the successful development of its whole, Mr. Cagara and Mr. Gabinete, the two developers of this program, have come to an agreement to disclose the said information as they believe that the program has reached its current state only through their efforts of working as a team. In fact, each block of codes present in the program has been made possible only because of the consistent synchronous communication shared by the two and that no block of codes have reached its final structure without applying the ideas both developers had in their minds.

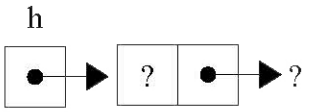
To simply say, no part of this program has been fully created with the efforts of one person alone; rather, each of the two has equally made a contribution (whether an approval for the consultation of the other or a direct modification of the code) for the finished structure of each block of codes present in this program. Thus, to be fair for each developer, so that at least none of them would have to be at a disadvantage position for the other has to indicate in the documentation part of the program that a particular part of code is his when it is actually a product of their teamwork, then the two decided to not follow this instruction in submitting c - exercises. Since both developers have made equal contributions to the entirety of this program and do have a complete understanding of how each block of codes work, the two wished to receive a mark uniform to them. They are more than happy to share the results of their fruitful work equally whether the mark that'd be bestowed upon them is excellent or not.

Pen-and-Paper Exercise

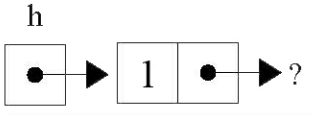
Now try it on your own and submit your answers to your lab instructor. Given the drawing conventions discussed earlier, draw and the effect of each of the assignment statements in the given sample code.

```
//Sample code for linked list
#include<stdio.h>

typedef struct nodetag{
  int x;
  struct nodetag *next;
}node;

int main(){
  node *h, *temp;

  //first node
  h=(node *)malloc(sizeof(node));
```

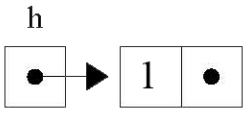




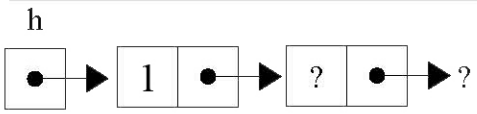

```
  h->x=1;
```



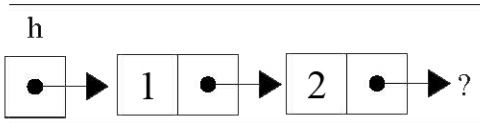

```
  h->next=NULL;
```



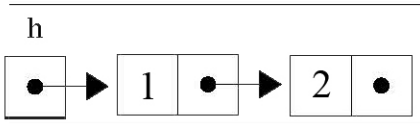

```
  //second node
  h->next=(node *)malloc(sizeof(node));
```



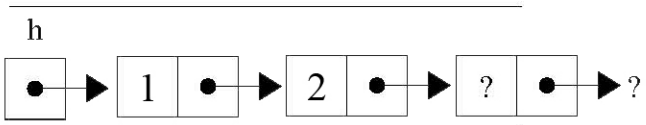

```
  h->next->x=2;
```



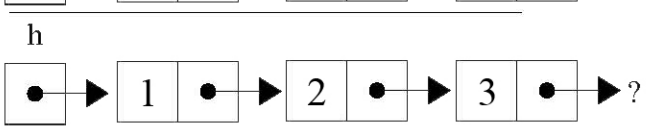

```
  h->next->next=NULL;
```



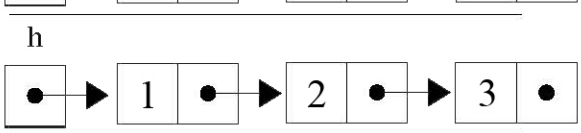

```
  //third node
  h->next->next=(node *)malloc(sizeof(node));
```




```
  h->next->next->x=3;
```

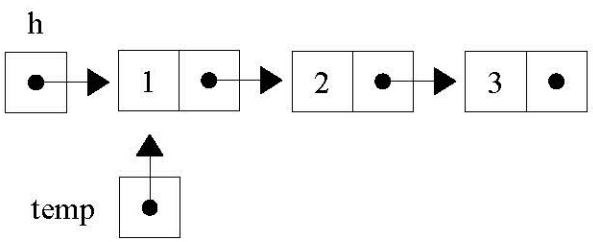



```
  h->next->next->next=NULL;
```

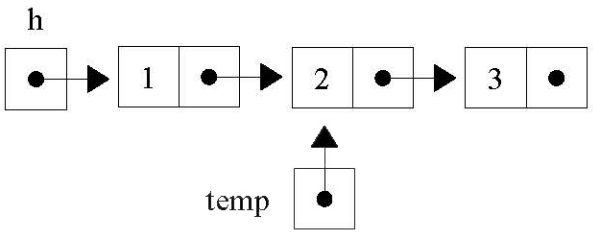



```
  //display the contents of the linked list

  temp=h;
  while(temp!=NULL){
      printf("%3i ",temp->x);
      temp=temp->next;
  }
  print("\n");
```

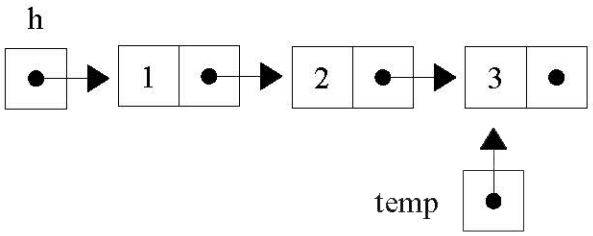
step 0: temp=h;



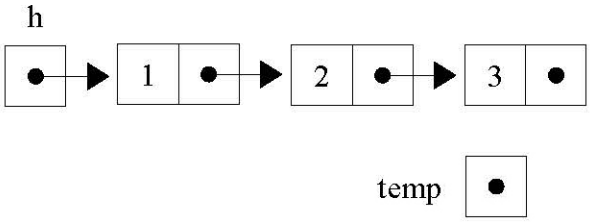
step 1: after (1st) temp=temp->next;



step 2: after (2nd) temp=temp->next;

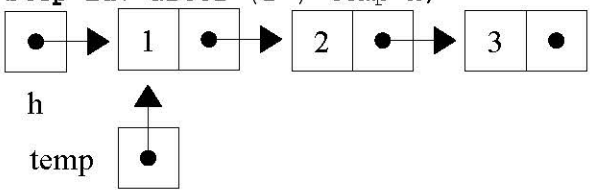


step 3: after (3rd) temp = temp->next;

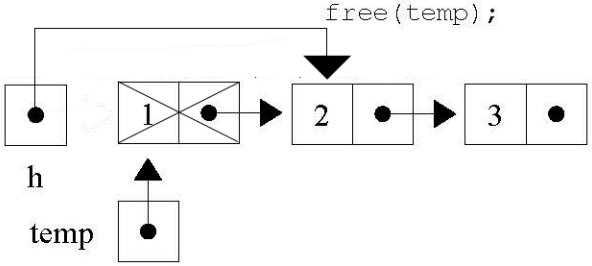



```
  //deallocation

  while(h!=NULL){
      temp=h;
      h=h->next;
      free(temp);
  }

  return(0);

}//end of main
```

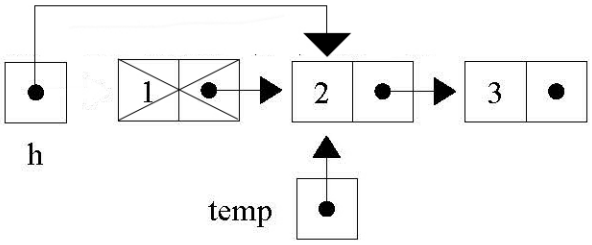
step 1a: after (1st) temp=h;



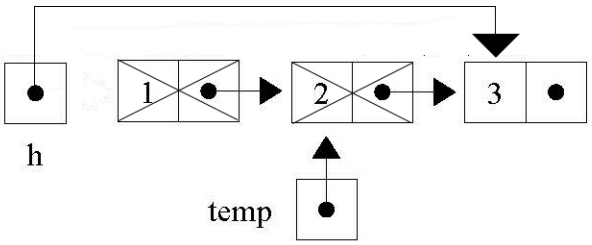
step 1b: after (1st) h=h->next;
free(temp);



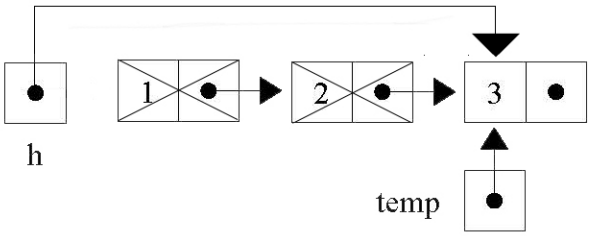
step 2a: after (2nd) temp=h;



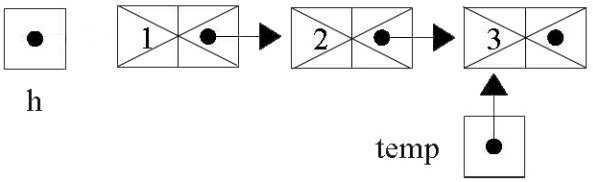
step 2b: after (2nd) h=h->next;
free(temp);



step 3a: after (3rd) temp=h;



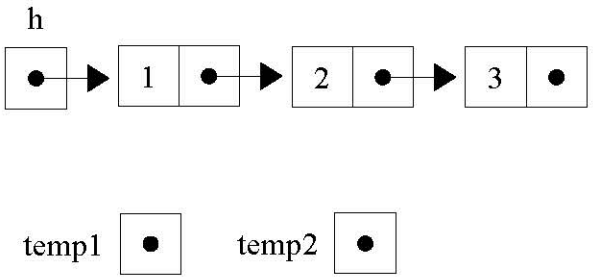
step 3b: after (3rd) h=h->next;
free(temp);



Now consider the code snippet below and trace it starting from the given initial setup. Go through the loop and draw the effect of **EACH** of the assignment, starting from the first **temp1=h;** assignment statement up to the last **temp2=temp1;** assignment statement and finally, the effect of **h=temp2;** assignment statement.

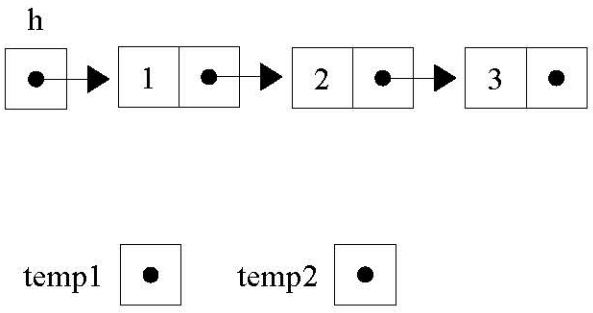
Hint: For this part you are encouraged to do this one drawing per page so it can be browsed like a "flip book".

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



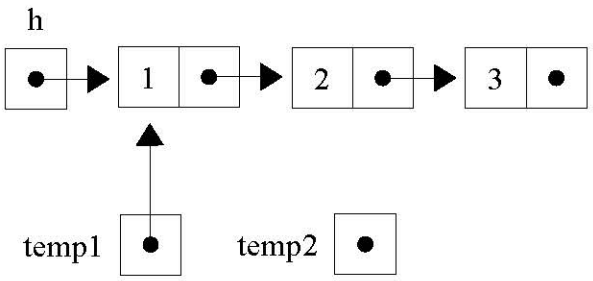
INITIAL SETUP

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



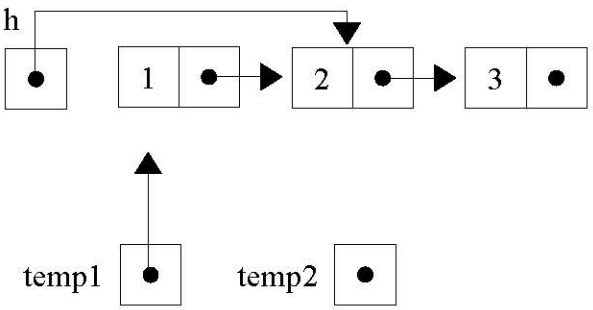
AFTER 1st temp1=h;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



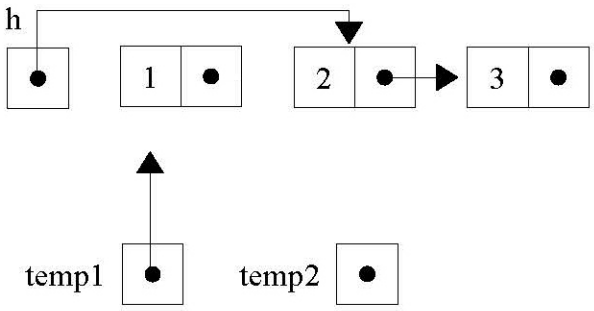
AFTER 1st h=h->next;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



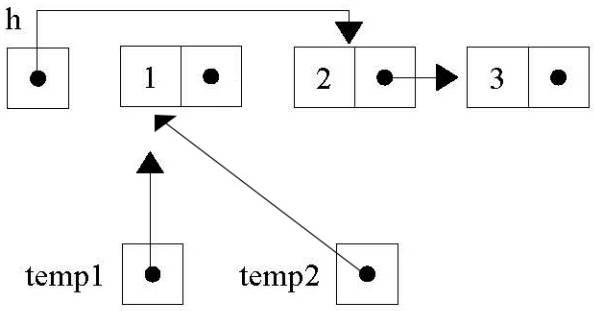
AFTER 1st temp1->next = temp2;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



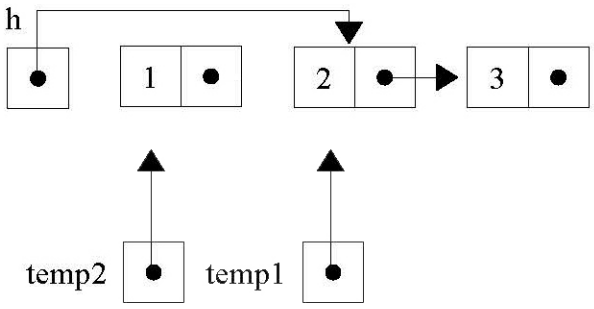
AFTER 1st temp2 = temp1;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



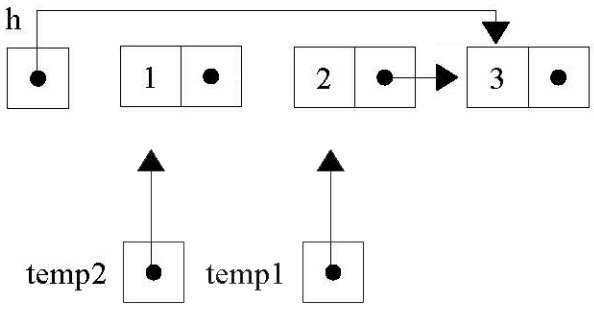
AFTER 2nd temp1 = h;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



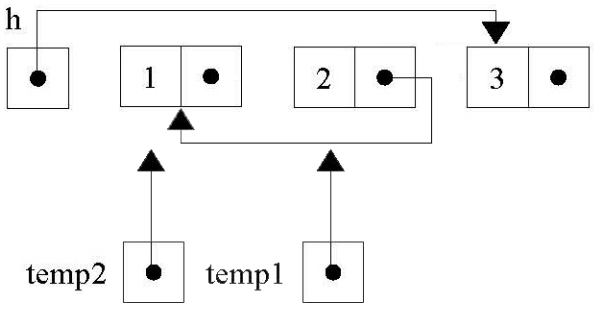
AFTER 2nd h = h->next;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



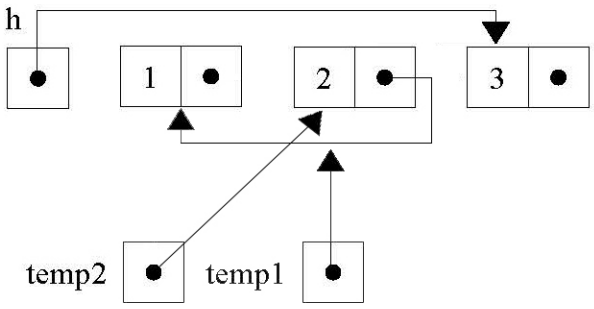
AFTER 2nd temp1->next = temp2;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



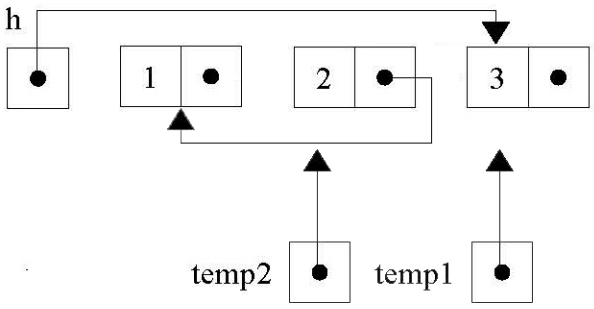
AFTER 2nd temp2 = temp1;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



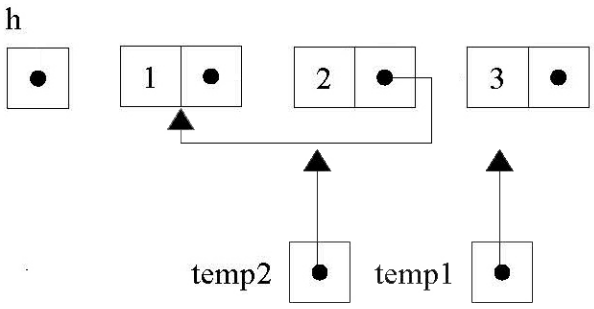
AFTER 3rd temp1 = h;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



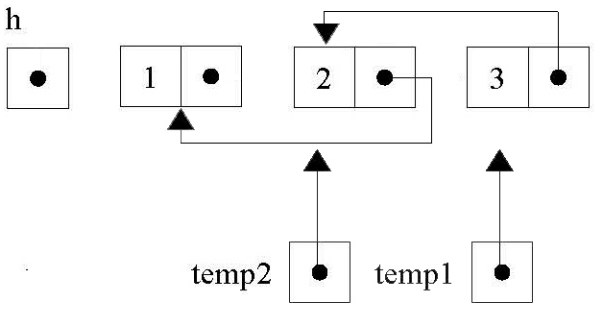
AFTER 3rd h = h->next;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



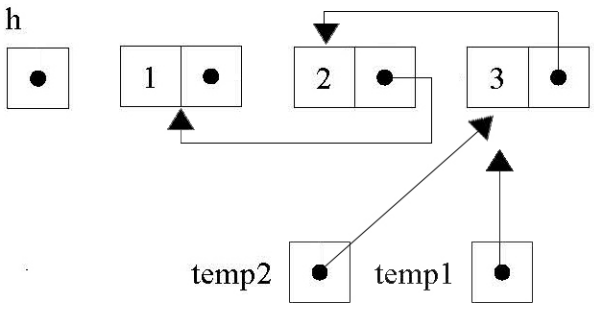
AFTER 3rd temp1->next = temp2;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



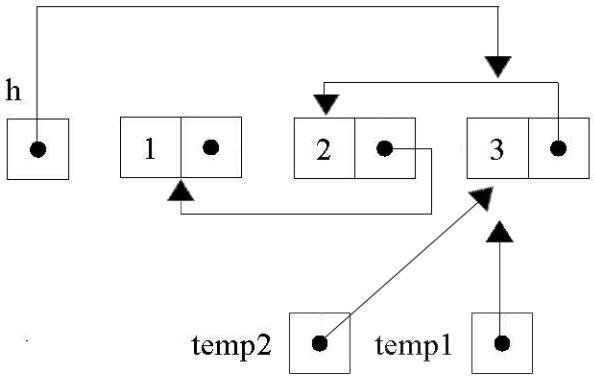
AFTER 3rd temp2 = temp1;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



AFTER h = temp2;

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```



Final diagram

```
//assume node *h; holds the initial list

node *temp1=NULL, *temp2=NULL;

while (h!=NULL){
      temp1=h;
      h=h->next;
      temp1->next=temp2;
      temp2=temp1;
}
h=temp2;
```