For the following questions, participants will be asked to complete common code comprehension tasks. They will be evaluated according to their correctness and time required to complete the tasks. This document contains the uncommented versions of these questions, but the code the participants are asked to read will be commented in different ways to experimentally determine the usefulness of the different comments.

The following code is an implementation of a linked list (only the functions that are necessary for the question are shown). It contains bugs in the reverse() and findMid() functions, which are shown by the actual and expected outputs. Locate each bug, fix it, and place the corrected code in the answer box.

```
#include <stdio.h> #include <string.h> #include <stdlib.h> #include <stdbool.h>

struct node {
  int data;
  int key;
  struct node *next;
};

struct node *head = NULL;

void printList() {
  struct node *ptr = head;
  printf("[ ");

  while(ptr != NULL) {
    printf("(%d,%d) ",ptr->key,ptr->data);
    ptr = ptr->next;
  }

  printf(" ]");
}

void insertFirst(int key, int data) {
  struct node *link = (struct node*) malloc(sizeof(struct node));

  link->key = key;
  link->data = data;
  link->next = head;
  head = link;
```

```
}

void reverse(struct node** head) {
   struct node* cur = *head;
   struct node* next = NULL;
   struct node* prev = NULL;

   while(cur!=NULL){
       cur->next = prev;
       next = cur->next;
       prev = cur;
       cur = next;
   }

   *head = prev;
}

struct node* findMid(){
    struct node* mid = head;
    struct node* cur = head;

    if(cur != NULL){
        cur = cur->next;
    }
    for(int i=0; cur!=NULL; ++i){
        cur = cur->next;
        if (i%2==0){
            mid = mid->next;
        }
    }

    return mid;
}

int main(){
    insertFirst(1,10);
    insertFirst(2,20);
    insertFirst(3,30);
    insertFirst(4,1);
    insertFirst(5,40);
    insertFirst(6,56);

    printf("Original List: \n");
    printList();

    printf("\n\nMid: \n");
    printf("Expected: 4");
    printf("\nActual: %d", findMid()->key);

    reverse(&head);

    printf("\n\nReversed List: \n");
    printf("Expected: [ (1,10) (2,20) (3,30) (4,1) (5,40) (6,56)  ]\n");
    printf("Actual: ");
    printList();
}
```

Output:

```
Original List:
[ (6,56) (5,40) (4,1) (3,30) (2,20) (1,10)  ]

Mid:
Expected: 4
Actual: 3

Reversed List:
Expected: [ (1,10) (2,20) (3,30) (4,1) (5,40) (6,56)  ]
Actual: [ (6,56)  ]
```

Answer:

```
void reverse(struct node** head) {
    /*Your code here*/
}

struct node* findMid(){
    /*Your code here*/
}
```