```
// ***
// *** You MUST modify this file
// ***

#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <string.h>
#include "hw12.h"

// DO NOT MODIFY this function --->>>
void printListNode(ListNode * head)
{
  ListNode * p = head;
  printf("printListNode: ");
  while (p != NULL)
    {
      printf("%7d ", p -> value);
      p = p -> next;
    }
  printf("\n");
}
// <<<--- until here

// You MUST modify the following functions

#ifdef TEST_CREATELIST
// create a linked list storing values 0, 1, 2, ... valn - 1
// The first node (head) stores 0, the next node stores 1,
// ..., the last node stores valn - 1
// return the head of the linked listn
ListNode * createList(int valn)
{
 ListNode * head = NULL;
 int i;
 for(i = (valn - 1); i >= 0; i--)
 {
  ListNode * nd = malloc(sizeof(ListNode));
  nd -> value = i;
  nd -> next = head;
  head = nd;
 }
 return head;
}
#endif

#ifdef TEST_ELIMINATE
// eliminate the nodes in the linked list
// starting from the head, move one node at a time and count to valk.
// eliminate that node, keep counting
```

```
//
// when reaching the end of the list, continue from the beginning of
// the list
//
// print the values of the nodes to be deleted
void eliminate(ListNode * head, int valk)
{
 //int valk_final = valk;
 //int valk_intial = 1;
 //int g = 0;
 //int counter = 0;
 //int list_count;
 //int k = 1;

 //while(ptr != NULL)
 //{
  //list_count++;
 //}

 int k = 1;
 ListNode * ptr = head; //points to head


 while(head -> next != NULL)
 {
  while(k < valk)
  {
   k++;
   if(ptr -> next == NULL)
   {
    ptr = head;
   }
   else
   {
    ptr = ptr -> next;
   }
  }//inside while

  if(k == valk)
  {
   head = deleteNode(head, ptr);
   //ListNode * n = ptr;
   //ptr = ptr -> next;
   //free(n);
   k = 0;
  }
 }//outside while

 printf("%d\n", head -> value);
 free(head);
```

```
}
#endif

#ifdef TEST_DELETENODE
// head points to the first node in the linked list
// todelete points  to the node to be deleted
//
// delete the node and return the head of the linked list
// release the memory of the deleted node
//
// should check several conditions:
// 1. If head is NULL, the list is empty and this function returns
//    NULL
// 2. If todelete is NULL, nothing can be deleted, return head
// 3. If todelete is not in the list, keep the list unchanged and
//    return head
// It is possible that todelete is the first node in the list (i.e.,
// the head). If this occurs, return the second node of the list.
ListNode * deleteNode(ListNode * head, ListNode * todelete)
{
 ListNode * pntr = head; //ptr to head
 int count = 0;
 ListNode * previous = NULL;

 if(pntr == NULL)
 {
  return NULL;
 }
 if(todelete == NULL)
 {
  return head;
 }


 while(pntr != NULL) //checks if todelete is in the list
 {
  if(todelete == pntr)
  {
  count++;
  }
  pntr = pntr -> next;
 }

 if(count == 0) //if not, return head
 {
  return head;
 }

 pntr = head;
```

```
 if(todelete == pntr) //if deleting the head
 {
  ListNode * p = pntr;
  head = pntr -> next;
  printListNode(p);
  printf("%d\n", p -> value);
  free(pntr);
  return head;
 }

 else //if not deleting the head
 {
  while(pntr != NULL)
  {
   if(todelete == pntr)
   {
    previous -> next = pntr -> next;
    ListNode * n = pntr;
    pntr = pntr -> next;
    printListNode(n);
    printf("%d\n", n -> value);
    free(n);
   }
   else
   {
    previous = pntr;
    pntr = pntr -> next;
   }
  }
 }

return(head);
}
#endif
```