|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 肖阳 | 学号 | 24300240055 | 第几次实验 | 15 |
| 实验代码（将代码粘贴在下面，如果有多题，将每一题的题目写在代码前面，加粗表示） | | | | | |
| #include <stdio.h>  #include <stdlib.h>  #include <assert.h>  // Node of a linked list  struct intNode {      int value;      struct intNode\* next;  };  // Create a node, given its value and next, return its pointer.  // The caller is responsible for freeing the node.  struct intNode\* create\_node(int val, struct intNode\* next) {      struct intNode\* new\_node;      new\_node = (struct intNode\*) malloc(sizeof(struct intNode));      new\_node->value = val;      new\_node->next = next;      return new\_node;  }  // Given an array of integers of length 'size', create a linked  // list (of length size+1).  //  // Return the empty header node (with value = 0).  struct intNode\* create\_linked\_list(int\* list, int size) {      struct intNode\* head = create\_node(0, NULL); // Create header node      if (size == 0) return head;        struct intNode\* current = head;      for (int i = 0; i < size; i++) {          current->next = create\_node(list[i], NULL);          current = current->next;      }      return head;  }  // Traverse the linked list and store its values in an array  // Skip the empty header node.  void traverse\_linked\_list(struct intNode\* head, int\* list, int\* size) {      \*size = 0;      struct intNode\* current = head->next; // Skip header node      while (current != NULL) {          list[\*size] = current->value;          (\*size)++;          current = current->next;      }  }  // Free the entire linked list, given the header node.  void free\_linked\_list(struct intNode\* head) {      struct intNode\* current = head;      while (current != NULL) {          struct intNode\* next = current->next;          free(current);          current = next;      }  }  // Search for the first node in the linked list with the given  // target value after the given node 'head'.  // If target value is not found, return an empty pointer.  struct intNode\* search(struct intNode\* head, int target\_value) {      struct intNode\* current = head;      while (current != NULL) {          if (current->value == target\_value) {              return current;          }          current = current->next;      }      return NULL;  }  // Given a node 'prev', insert the 'current' node right after  // 'prev'.  void insert(struct intNode\* prev, struct intNode\* current) {      current->next = prev->next;      prev->next = current;  }  // Delete the current node given its previous node.  void delete(struct intNode\* prev, struct intNode\* current) {      prev->next = current->next;      free(current);  }  // Test function for assertions  void test\_linked\_list() {      // Test create\_linked\_list and traverse\_linked\_list      int arr[] = {1, 2, 3, 4, 5};      struct intNode\* list = create\_linked\_list(arr, 5);      int output[5];      int size;      traverse\_linked\_list(list, output, &size);      assert(size == 5);  // Ensure the list has 5 elements      for (int i = 0; i < size; i++) {          assert(output[i] == arr[i]);  // Ensure the values match the original array      }      // Test find function      struct intNode\* found = search(list, 3);      assert(found != NULL);  // Node with value 3 should be found      assert(found->value == 3);  // Ensure the value is correct      found = search(list, 6);      assert(found == NULL);  // Node with value 6 should not be found      // Test insert function      struct intNode\* new\_node = create\_node(99, NULL);      insert(list->next, new\_node);  // Insert 99 after the first node (1)      assert(list->next->next->value == 99);  // Ensure 99 is correctly inserted after 1      // Test delete function      delete(list->next, new\_node);  // Delete the node with value 99      assert(list->next->next->value == 2);  // Ensure 99 was deleted      // Test free\_linked\_list      free\_linked\_list(list);  // Ensure the memory is freed without memory leaks  }  int main() {      test\_linked\_list();  // Run the tests      printf("All tests passed!\n");      return 0;  } | | | | | |
| 运行结果（截图运行结果，如果有多题，写清题目加粗表示，再粘贴截图） | | | | | |
|  | | | | | |