```
#include <stdio.h>
#include <stdlib.h>
#include "linklist.h"
linklist list_create() {
      linklist H;
      H = (linklist)malloc(sizeof(listnode));
      if (H == NULL) {
            printf("malloc failed\n");
            return H;
      }
      H->data = 0;
      H->next = NULL;
      return H;
}
int list_tail_insert(linklist H, data_t value) {
      linklist p;
      linklist q;
      if (H == NULL) {
    printf("H is NULL\n");
            return -1;
      }
      //1 new node p
      if ((p = (linklist)malloc(sizeof(listnode))) == NULL) {
            printf("malloc failed\n");
            return -1;
      p->data = value;
      p->next = NULL;
      //2 locate locate locate locate locate locate locate locate tail node
      q = H;
      while (q->next != NULL) {
            q = q->next;
      }
      //3 insert
      q - next = p;
      return 0;
}
linklist list_get(linklist H, int pos) {
      linklist p;
      int i;
      if (H == NULL) {
            printf("H is NULL\n");
            return NULL;
      if (pos == -1) {
            return H;
```

```
}
     if (pos < -1) {
            printf("pos is invalid\n");
            return NULL;
      }
      p = H;
      i = -1;
      while (i < pos) {
            p = p->next;
            if (p == NULL) {
                  printf("pos is invalid\n");
                  return NULL;
            i++;
      }
      return p;
}
int list_insert(linklist H, data_t value, int pos) {
      linklist p;
      linklist q;
      if (H == NULL) {
            printf("H is NULL\n");
            return -1;
      }
      //1 locate node p (pos-1)
      p = list_get(H, pos-1);
      if (p == NULL) {
            return -1;
      }
      //2 new node q
      if ((q = (linklist)malloc(sizeof(listnode))) == NULL) {
            printf("malloc failed\n");
            return -1;
      }
      q->data = value;
      q->next = NULL;
      //3 insert
      q->next = p->next;
      p->next = q;
      return 0;
}
int list_delete(linklist H, int pos) {
      linklist p;
      linklist q;
      //1
      if (H == NULL) {
            printf("H is NULL\n");
            return -1;
```

```
}
      //2 locate prior
      p = list_get(H, pos-1);
      if (p == NULL)
            return -1;
      if (p->next == NULL) {
            printf("delete pos is invalid\n");
            return -1;
      }
      //3 update list
      q = p->next;
      p->next = q->next;//p->next = p->next->next;
      //4 free
      printf("free:%d\n", q->data);
      free(q);
      q = NULL;
      return 0;
}
int list_show(linklist H) {
      linklist p;
     if (H == NULL) {
            printf("H is NULL\n");
            return -1;
      }
      p = H;
      while (p->next != NULL) {
            printf("%d ", p->next->data);
            p = p->next;
      puts("");
      return 0;
}
linklist list_free(linklist H) {
      linklist p;
      if (H == NULL)
            return NULL;
      p = H;
      printf("free:");
      while (H != NULL) {
            p = H;
            printf("%d ", p->data);
            free(p);
            H = H->next;
      puts("");
```

```
return NULL;
}
```