#include “lists.h”

/\*\*

\* add\_nodeint – adds a node at the end

\* @head: start of node

\* @n: value to give new node

\* Return: the linked list

\*/

Listint\_t \*add\_nodeint(listint\_t \*\*head, const int n)

{

Listint\_t \*new;

New = malloc(sizeof(listint\_t));

If (new == NULL)

Return (NULL);

New->n = n;

New->next = \*head;

\*head = new;

Return (\*head);

}

#ifndef LISTS\_H

#define LISTS\_H

#include <stddef.h>

#include <stdlib.h>

#include <stdio.h>

int \_putchar(char c);

/\*\*

\* struct list\_s - singly linked list

\* @str: string - (malloc'ed string)

\* @len: length of the string

\* @next: points to the next node

\*

\* Description: singly linked list node structure

\* for Holberton project

\*/

typedef struct list\_s

{

char \*str;

unsigned int len;

struct list\_s \*next;

} list\_t;

/\*\*

\* struct listint\_s - singly linked list

\* @n: integer

\* @next: points to the next node

\*

\* Description: singly linked list node structure

\* for Holberton project

\*/

typedef struct listint\_s

{

int n;

struct listint\_s \*next;

} listint\_t;

size\_t print\_list(const list\_t \*h);

size\_t list\_len(const list\_t \*h);

list\_t \*add\_node(list\_t \*\*head, const char \*str);

list\_t \*add\_node\_end(list\_t \*\*head, const char \*str);

void free\_list(list\_t \*head);

size\_t print\_listint(const listint\_t \*h);

size\_t listint\_len(const listint\_t \*h);

listint\_t \*add\_nodeint(listint\_t \*\*head, const int n);

listint\_t \*add\_nodeint\_end(listint\_t \*\*head, const int n);

void free\_listint(listint\_t \*head);

void free\_listint2(listint\_t \*\*head);

int pop\_listint(listint\_t \*\*head);

listint\_t \*get\_nodeint\_at\_index(listint\_t \*head, unsigned int index);

int sum\_listint(listint\_t \*head);

listint\_t \*insert\_nodeint\_at\_index(listint\_t \*\*head, unsigned int idx, int n);

int delete\_nodeint\_at\_index(listint\_t \*\*head, unsigned int index);

#endif