(/)

Curriculum

SE Foundations Average: 108.76%

0x17. C - Doubly linked lists

C Algorithm Data structure

By: Julien Barbier

Weight: 1

An auto review will be launched at the deadline

In a nutshell...

• Auto QA review: 86.0/98 mandatory & 0.0/16 optional

• Altogether: 87.76%

Mandatory: 87.76%Optional: 0.0%

o Calculation: 87.76% + (87.76% * 0.0%) == 87.76%

Resources

Read or watch:

What is a Doubly Linked List (/rltoken/C5_IRM981SVn8oA8RP3gag)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/0ABh2M07w5kdsh9gRx1XwA), without the help of Google:

General



What is a doubly linked list

How to use doubly linked lists

Help

- Understand and know how to implement the various operations (deletion, insertion, etc) with doubly
 (/) linked lists
 - Start to look for the right source of information without too much help

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: vi, vim, emacs
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using python3 (version 3.8.5)
- All your files should end with a new line
- A README.md file, at the root of the folder of the project is mandatory
- Your code should use the Betty style. It will be checked using betty-style.pl (https://github.com/alx-tools/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/alx-tools/Betty/blob/master/betty-doc.pl)
- You are not allowed to use global variables
- No more than 5 functions per file
- The only C standard library functions allowed are malloc, free, printf and exit
- In the following examples, the main.c files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account). We will use our own main.c files at compilation. Our main.c files might be different from the one shown in the examples
- The prototypes of all your functions should be included in your header file called lists.h
- Don't forget to push your header file
- All your header files should be include guarded

More Info

Please use this data structure for this project:

```
# struct dlistint_s - doubly linked list

* @n: integer
 * @prev: points to the previous node
 * @next: points to the next node

*
 * Description: doubly linked list node structure

*
 */
typedef struct dlistint_s
{
  int n;
  struct dlistint_s *prev;
  struct dlistint_s *next;
} dlistint_t;
```

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Show quiz)

Tasks

0. Print list

mandatory

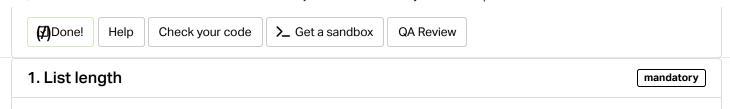
Score: 100.0% (Checks completed: 100.0%)

Write a function that prints all the elements of a dlistint_t list.

- Prototype: size_t print_dlistint(const dlistint_t *h);
- Return: the number of nodes
- Format: see example

```
المِينِ lien@ubuntu:~/0x17. Doubly linked lists$ cat 0-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
    dlistint_t *head;
    dlistint_t *new;
    dlistint_t hello = {8, NULL, NULL};
    size_t n;
    head = &hello;
    new = malloc(sizeof(dlistint_t));
    if (new == NULL)
    {
        dprintf(2, "Error: Can't malloc\n");
        return (EXIT_FAILURE);
    }
    new->n = 9;
    head->prev = new;
    new->next = head;
    new->prev = NULL;
    head = new;
    n = print_dlistint(head);
    printf("-> %lu elements\n", n);
    free(new);
    return (EXIT_SUCCESS);
}
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 0-main.c 0-print_dlistint.c -o a
julien@ubuntu:~/0x17. Doubly linked lists$ ./a
9
-> 2 elements
julien@ubuntu:~/0x17. Doubly linked lists$
```

- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists
- File: 0-print_dlistint.c



Write a function that returns the number of elements in a linked $\mbox{dlistint_t}$ list.

• Prototype: size_t dlistint_len(const dlistint_t *h);

Score: 100.0% (Checks completed: 100.0%)

```
المِينِ lien@ubuntu:~/0x17. Doubly linked lists$ cat 1-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
    dlistint_t *head;
    dlistint_t *new;
    dlistint_t hello = {8, NULL, NULL};
    size_t n;
    head = &hello;
    new = malloc(sizeof(dlistint_t));
    if (new == NULL)
    {
        dprintf(2, "Error: Can't malloc\n");
        return (EXIT_FAILURE);
    }
    new->n = 9;
    head->prev = new;
    new->next = head;
    new->prev = NULL;
    head = new;
    n = dlistint_len(head);
    printf("-> %lu elements\n", n);
    free(new);
    return (EXIT_SUCCESS);
}
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 1-main.c 1-dlistint_len.c -o b
julien@ubuntu:~/0x17. Doubly linked lists$ ./b
-> 2 elements
julien@ubuntu:~/0x17. Doubly linked lists$
```

- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists
- File: 1-dlistint_len.c

Q

2₍/Add node

mandatory

```
Score: 100.0% (Checks completed: 100.0%)
```

Write a function that adds a new node at the beginning of a dlistint_t list.

- Prototype: dlistint_t *add_dnodeint(dlistint_t **head, const int n);
- Return: the address of the new element, or NULL if it failed

```
julien@ubuntu:~/0x17. Doubly linked lists$ cat 2-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
/**
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
    dlistint_t *head;
    head = NULL;
    add_dnodeint(&head, 0);
    add_dnodeint(&head, 1);
    add_dnodeint(&head, 2);
    add_dnodeint(&head, 3);
    add_dnodeint(&head, 4);
    add_dnodeint(&head, 98);
    add_dnodeint(&head, 402);
    add_dnodeint(&head, 1024);
    print_dlistint(head);
    return (EXIT_SUCCESS);
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 2-main.c 2-add_dnodeint.c 0-print_dlistint.c -o c
julien@ubuntu:~/0x17. Doubly linked lists$ ./c
1024
402
98
4
3
2
1
julien@ubuntu:~/0x17. Doubly linked lists$
```

- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists
- File: 2-add_dnodeint.c

3. Add node at the end

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a function that adds a new node at the end of a dlistint_t list.

- Prototype: dlistint_t *add_dnodeint_end(dlistint_t **head, const int n);
- Return: the address of the new element, or NULL if it failed

```
jµlien@ubuntu:~/0x17. Doubly linked lists$ cat 3-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
    dlistint_t *head;
    head = NULL;
    add_dnodeint_end(&head, 0);
    add_dnodeint_end(&head, 1);
    add_dnodeint_end(&head, 2);
    add_dnodeint_end(&head, 3);
    add_dnodeint_end(&head, 4);
    add_dnodeint_end(&head, 98);
    add_dnodeint_end(&head, 402);
    add_dnodeint_end(&head, 1024);
    print_dlistint(head);
    return (EXIT_SUCCESS);
}
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 3-main.c 3-add_dnodeint_end.c 0-print_dlistint.c -o d
julien@ubuntu:~/0x17. Doubly linked lists$ ./d
0
1
2
3
4
98
402
1024
julien@ubuntu:~/0x17. Doubly linked lists$
```

• GitHub repository: alx-low_level_programming

Check your code

- Directory: 0x17-doubly_linked_lists
- File: 3-add_dnodeint_end.c

>_ Get a sandbox QA Review

☑ Done!

Help

4₍/Free list

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a function that frees a dlistint_t list.

• Prototype: void free_dlistint(dlistint_t *head);

```
المِينِ lien@ubuntu:~/0x17. Doubly linked lists$ cat 4-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
    dlistint_t *head;
    head = NULL;
    add_dnodeint_end(&head, 0);
    add_dnodeint_end(&head, 1);
    add_dnodeint_end(&head, 2);
    add_dnodeint_end(&head, 3);
    add_dnodeint_end(&head, 4);
    add_dnodeint_end(&head, 98);
    add_dnodeint_end(&head, 402);
    add_dnodeint_end(&head, 1024);
    print_dlistint(head);
    free_dlistint(head);
    head = NULL;
    return (EXIT_SUCCESS);
}
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 4-main.c 3-add_dnodeint_end.c 0-print_dlistint.c 4-free_dlistint.c -o e
julien@ubuntu:~/0x17. Doubly linked lists$ valgrind ./e
==4197== Memcheck, a memory error detector
==4197== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==4197== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==4197== Command: ./e
==4197==
0
1
2
3
4
98
402
1024
==4197==
==4197== HEAP SUMMARY:
==4197==
             in use at exit: 0 bytes in 0 blocks
           total heap usage: 9 allocs, 9 frees, 1,216 bytes allocated
==4197==
==4197==
==4197== All heap blocks were freed -- no leaks are possible
==4197==
```

==4197== For counts of detected and suppressed errors, rerun with: -v #\ddf4197== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0) julien@ubuntu:~/0x17. Doubly linked lists\$

Repo:

- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists
- File: 4-free_dlistint.c

5. Get node at index

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a function that returns the nth node of a dlistint_t linked list.

- Prototype: dlistint_t *get_dnodeint_at_index(dlistint_t *head, unsigned int index);
- where index is the index of the node, starting from 0
- if the node does not exist, return NULL

```
jµlien@ubuntu:~/0x17. Doubly linked lists$ cat 5-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
    dlistint_t *head;
    dlistint_t *node;
    head = NULL;
    add_dnodeint_end(&head, 0);
    add_dnodeint_end(&head, 1);
    add_dnodeint_end(&head, 2);
    add_dnodeint_end(&head, 3);
    add_dnodeint_end(&head, 4);
    add_dnodeint_end(&head, 98);
    add_dnodeint_end(&head, 402);
    add_dnodeint_end(&head, 1024);
    print_dlistint(head);
    node = get_dnodeint_at_index(head, 5);
    printf("%d\n", node->n);
    free_dlistint(head);
    head = NULL;
    return (EXIT_SUCCESS);
}
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 5-main.c 3-add_dnodeint_end.c 0-print_dlistint.c 4-free_dlistint.c 5-get_dnode
int.c -o h
julien@ubuntu:~/0x17. Doubly linked lists$ ./h
0
1
2
3
4
98
402
1024
julien@ubuntu:~/0x17. Doubly linked lists$
```

GitHub repository: alx-low_level_programming

3/6/24, 9:56 AM Project: 0x17. C - Doubly linked lists | Nairobi Intranet • Directory: 0x17-doubly_linked_lists (/) File: 5-get_dnodeint.c ☑ Done! >_ Get a sandbox Help Check your code **QA Review** 6. Sum list mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a function that returns the sum of all the data (n) of a dlistint_t linked list.

- Prototype: int sum_dlistint(dlistint_t *head);
- if the list is empty, return 0

```
jµlien@ubuntu:~/0x17. Doubly linked lists$ cat 6-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
    dlistint_t *head;
    int sum;
    head = NULL;
    add_dnodeint_end(&head, 0);
    add_dnodeint_end(&head, 1);
    add_dnodeint_end(&head, 2);
    add_dnodeint_end(&head, 3);
    add_dnodeint_end(&head, 4);
    add_dnodeint_end(&head, 98);
    add_dnodeint_end(&head, 402);
    add_dnodeint_end(&head, 1024);
    sum = sum_dlistint(head);
    printf("sum = %d\n", sum);
    free_dlistint(head);
    head = NULL;
    return (EXIT_SUCCESS);
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra 6-mai
n.c -std=qnu89 3-add_dnodeint_end.c 4-free_dlistint.c 6-sum_dlistint.c -o i
julien@ubuntu:~/0x17. Doubly linked lists$ ./i
sum = 1534
julien@ubuntu:~/0x17. Doubly linked lists$
```

- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists
- File: 6-sum_dlistint.c

☑ Done! Help Check your code ➤ Get a sandbox QA Review

7. Insert at index

0 - - - - 0 00/ / 0/

mandatory

SCOre: U.U% (Checks completed: U.U%)

(/)

Write a function that inserts a new node at a given position.

- Prototype: dlistint_t *insert_dnodeint_at_index(dlistint_t **h, unsigned int idx, int n);
- where idx is the index of the list where the new node should be added. Index starts at 0
- Returns: the address of the new node, or NULL if it failed
- if it is not possible to add the new node at index idx, do not add the new node and return NULL

Your files 2-add_dnodeint.c and 3-add_dnodeint_end.c will be compiled during the correction

```
jµlien@ubuntu:~/0x17. Doubly linked lists$ cat 7-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
   dlistint_t *head;
    head = NULL;
    add_dnodeint_end(&head, 0);
    add_dnodeint_end(&head, 1);
    add_dnodeint_end(&head, 2);
    add_dnodeint_end(&head, 3);
    add_dnodeint_end(&head, 4);
    add_dnodeint_end(&head, 98);
    add_dnodeint_end(&head, 402);
    add_dnodeint_end(&head, 1024);
    print_dlistint(head);
    printf("-----\n");
    insert_dnodeint_at_index(&head, 5, 4096);
    print_dlistint(head);
    free_dlistint(head);
    head = NULL;
    return (EXIT_SUCCESS);
}
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 7-main.c 2-add_dnodeint.c 3-add_dnodeint_end.c 0-print_dlistint.c 4-free_dlist
int.c 7-insert_dnodeint.c -o j
julien@ubuntu:~/0x17. Doubly linked lists$ ./j
0
1
2
3
4
98
402
1024
-----
1
2
3
4
4096
98
```

402 **(/)**24

julien@ubuntu:~/0x17. Doubly linked lists\$

Repo:

- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists
- File: 7-insert_dnodeint.c, 2-add_dnodeint.c, 3-add_dnodeint_end.c

□ Done? Help Check your code Ask for a new correction > Get a sandbox QA Review

8. Delete at index

mandatory

Score: 100.0% (Checks completed: 100.0%)

Write a function that deletes the node at index index of a dlistint_t linked list.

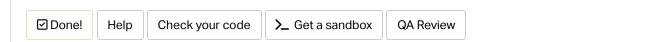
- Prototype: int delete_dnodeint_at_index(dlistint_t **head, unsigned int index);
- where index is the index of the node that should be deleted. Index starts at 0
- Returns: 1 if it succeeded, -1 if it failed

```
المِرَانِ lien@ubuntu:~/0x17. Doubly linked lists$ cat 8-main.c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "lists.h"
 * main - check the code
 * Return: Always EXIT_SUCCESS.
int main(void)
{
   dlistint_t *head;
   head = NULL;
   add_dnodeint_end(&head, 0);
   add_dnodeint_end(&head, 1);
   add_dnodeint_end(&head, 2);
   add_dnodeint_end(&head, 3);
   add_dnodeint_end(&head, 4);
   add_dnodeint_end(&head, 98);
   add_dnodeint_end(&head, 402);
   add_dnodeint_end(&head, 1024);
   print_dlistint(head);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 5);
   print_dlistint(head);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
   print_dlistint(head);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
   print_dlistint(head);
   printf("----\n");
   delete_dnodeint_at_index(&head, 0);
   print_dlistint(head);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
   print_dlistint(head);
   printf("----\n");
   delete_dnodeint_at_index(&head, 0);
   print_dlistint(head);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
   print_dlistint(head);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
```

```
printf("-----\n");
(/) delete_dnodeint_at_index(&head, 0);
   printf("-----\n");
   delete_dnodeint_at_index(&head, 0);
   print_dlistint(head);
   return (0);
}
julien@ubuntu:~/0x17. Doubly linked lists$ gcc -Wall -pedantic -Werror -Wextra -std=
gnu89 8-main.c 3-add_dnodeint_end.c 0-print_dlistint.c 4-free_dlistint.c 8-delete_dn
odeint.c -o k
julien@ubuntu:~/0x17. Doubly linked lists$ ./k
1
2
3
4
98
402
1024
-----
1
2
3
402
1024
-----
2
3
4
402
1024
-----
3
4
402
1024
-----
3
```

4
(∤) 2
1024
4
402
1024
402
1024
1024
julien@ubuntu:~/0x17. Doubly linked lists\$
Just tellen and the company states of the company o

- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists
- File: 8-delete_dnodeint.c



9. Crackme4

#advanced

Score: 0.0% (Checks completed: 0.0%)

Find the password for crackme4 (https://github.com/alx-tools/0x17.c).

- Save the password in the file 100-password
- Your file should contain the exact password, no new line, no extra space
- Hint: The program prints "OK" when the password is correct

Repo:



- GitHub repository: alx-low_level_programming
- Directory: 0x17-doubly_linked_lists

☐ Done?

Help

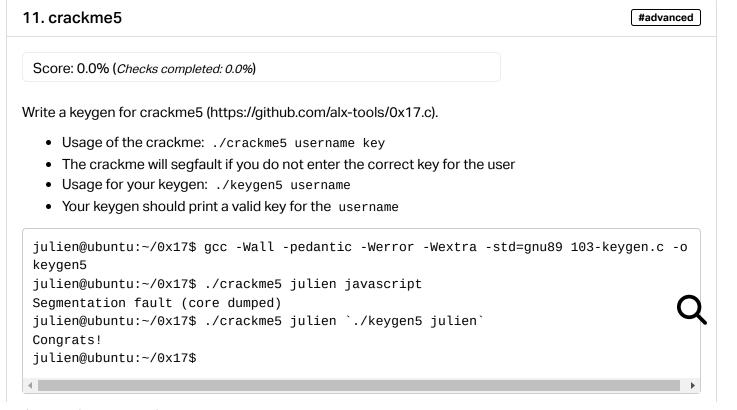
Check your code

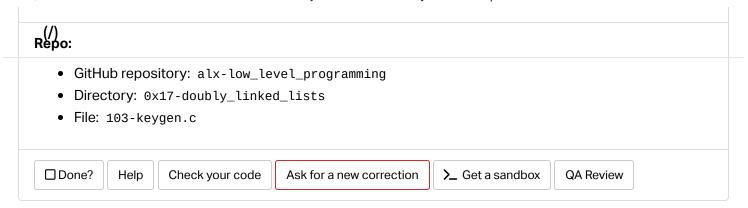
• File: 100-password (/)☐ Done? Help Check your code Ask for a new correction >_ Get a sandbox **QA Review** 10. Palindromes #advanced Score: 0.0% (Checks completed: 0.0%) A palindromic number reads the same both ways. The largest palindrome made from the product of two 2digit numbers is $9009 = 91 \times 99$. Find the largest palindrome made from the product of two 3-digit numbers. • Save the result in the file 102-result • Your file should contain the exact result, no new line, no extra space Repo: GitHub repository: alx-low_level_programming • Directory: 0x17-doubly_linked_lists • File: 102-result

Ask for a new correction

>_ Get a sandbox

QA Review





Copyright © 2024 ALX, All rights reserved.