

“slltool” Module’s Documentation

Description

“slltool” stands for “Singly Linked List Tool”. The module consists of several functions which are needed to implement a basic singly linked list (from now – linked list) and to access its information. The file supports a “generic linked list” and thus can work with any data type given to the data pointer, however, not all data types can be processed (see – [printList](#), [getElementValue](#)).

Note: The module does not account for error handling (except errors mentioned in function descriptions). Anything beyond the scope of this documentation may result in undefined behavior. It is the user’s responsibility to use the functions correctly. Module’s original functions are made with the intention to use them in accordance with this documentation only.

Additional functions and variables may be added by the user in the .h or .c files to change or improve the functionality and error handling of the module. **WARNING!** – the creator of this module is not responsible for any crashes, errors, bugs, or any sort of inconveniences caused by module’s original or user-defined files. Consider this a final warning – use and change the library at your own discretion!

Module uses standard libraries: <stdio.h>, <stdlib.h>, <string.h>, <stddef.h>

Structures:

Node	A struct to create a linked list
----------------------	----------------------------------

Functions:

Available to the user:

createList	Creates a linked list
deleteList	Deletes a linked list
printList	Prints all values from a linked list
insertElement	Inserts a value into a linked list
deleteElement	Deletes a node from a linked list
getListSize	Returns the size of a linked list
getElementValue	Returns the address of a linked list value
print[data-type]	Helps to execute the printList function

Hidden from the user:

checkMemAlloc	Checks variable’s memory allocation
checkIndex(GEV)	Checks correctness of an index

Node

Singly linked list structure

Structure is used to make a linked list and operate with its components.

The structure contains 2 members which are:

Member	Type	Meaning
data	void*	Stores data at a given node of a linked list
next	Node*	Points to the next node in a linked list

[Back to top](#)

createList

```
void createList(Node **head);
```

Creates a singly linked list

This function initializes the head variable of a linked list to *NULL* (which points to the first element of a linked list) thus creating a linked list.

Function does not check if the head variable has been initialized as a linked list. Responsibility to check this is given to the user.

Parameters

head Address of a pointer which will hold the address to the first element of a linked list.

Return value

none.

Example

```
1  #include <stdio.h>
2  #include "slltool.h"
3
4  int main ()
5  {
6      //Pointer to hold the address of the first linked list element
7      Node *first;
8
9      //Initialize the first element and create a linked list
10     createList(&first);
11
12     return 0;
13 }
```

[Back to top](#)

deleteList

```
void deleteList(Node **head);
```

Deletes a singly linked list

This function deletes all nodes from a linked list in turn effectively erasing the list. The head variable of the list is set to *NULL* to signify emptiness. Elements are deallocated using `<stdlib.h>` function *free*.

Function does not check if the head variable has been initialized as a linked list. Responsibility to check this is given to the user.

Parameters

head Address of pointer which holds the address to the first element of a linked list.

Return value

none.

Example

Output:

<pre>1 #include <stdio.h> 2 #include "slltool.h" 3 4 int main () 5 { 6 Node *head; 7 int num = 2; 8 9 createList(&head); 10 11 insertElement(&head, 1, &num, sizeof(int)); 12 insertElement(&head, 2, &num, sizeof(int)); 13 14 printf("Size before: %d\n", getListSize(&head)); 15 deleteList(&head); 16 printf("Size after: %d\n", getListSize(&head)); 17 18 return 0; 19 }</pre>	<pre>Size before: 2 Size after: 0</pre>
--	---

[Back to top](#)

printList

```
void printList(Node **head, void (*callPrint)(void *));
```

Prints all values of a linked list

This function prints out all values from a linked list which are not stored in derived data types (to print out elements from derived data types see [getElementValue](#)). Function uses additional functions from *print[data-type]* to be able to work with any primary data type.

Function does not check if the head variable has been initialized as a linked list. Responsibility to check this is given to the user.

Parameters

head Address of a pointer which holds the address to the first element of a linked list.

callPrint Address of a function from the [print\[data-type\]](#) function list to print the corresponding data type.

Return value

none.

Example

Output:

<pre>1 #include <stdio.h> 2 #include "slltool.h" 3 4 int main () 5 { 6 Node *head; 7 int num = 2; 8 9 createList(&head); 10 11 insertElement(&head, 1, &num, sizeof(int)); 12 insertElement(&head, 2, &num, sizeof(int)); 13 insertElement(&head, 3, &num, sizeof(int)); 14 15 printf("Elements of a linked list:\n"); 16 printList(&head, printInt); 17 18 return 0; 19 }</pre>	<pre>Elements of a linked list: 2 2 2</pre>
--	---

[Back to top](#)

insertElement

```
void insertElement(Node **head, int index, void*value, size_t value_size);
```

Inserts a value into a singly linked list

This function creates a linked list node and inserts a value in that node. Can be used to insert derived data types as well. The function checks if the `index` is less than 1 and if memory for the new node can be allocated.

Function does not check if the head variable has been initialized as a linked list. Responsibility to check this is given to the user.

Parameters

head	Address of a pointer which holds the address to the first element of a linked list.
index	Position at which the value is to be inserted.
value	A pointer to a value which is to be inserted.
value_size	Size of the value's data type.

Return value

none.

Example

Output:

<pre>1 #include <stdio.h> 2 #include "slltool.h" 3 4 int main () 5 { 6 Node *head; 7 int num = 1, jim = 2; 8 9 createList(&head); 10 11 insertElement(&head, 1, &num, sizeof(int)); 12 printf("First insert:\n"); 13 printList(&head, printInt); 14 insertElement(&head, 1, &jim, sizeof(int)); 15 printf("Second insert:\n"); 16 printList(&head, printInt); 17 18 return 0; 19 }</pre>	<pre>First insert: 1 Second insert: 2 1</pre>
--	---

deleteElement

```
void deleteElement(Node **head, int index)
```

Deletes a node from a singly linked list

This function deletes a single node from a linked list thus deleting all values that may be held in the node. The function checks if `index` is less than 1. Elements are deallocated using `<stdlib.h>` function *free*.

Function does not check if the head variable has been initialized as a linked list. Responsibility to check this is transferred to the user.

Parameters

`head` Address of a pointer which holds the address to the first element of a linked list.

`index` Position of the node which is to be deleted.

Return value

none.

Example

Output:

<pre>1 #include <stdio.h> 2 #include "slltool.h" 3 4 int main () 5 { 6 Node *head; 7 int num = 1, jim = 2; 8 9 createList(&head); 10 11 insertElement(&head, 1, &num, sizeof(int)); 12 insertElement(&head, 2, &jim, sizeof(int)); 13 printf("List before:\n"); 14 printList(&head, printInt); 15 deleteElement(&head, 2); 16 printf("List after: \n"); 17 printList(&head, printInt); 18 19 return 0; 20 }</pre>	<pre>List before: 1 2 List after: 1</pre>
---	---

[Back to top](#)

getListSize

```
size_t getListSize(Node **head);
```

Returns the size of a singly linked list

This function returns the size of an entire linked list. Failure to return the size results in undefined behavior.

Function does not check if the head variable has been initialized as a linked list. Responsibility to check this is transferred to the user.

Parameters

head Address of a pointer which holds the address to the first element of a linked list.

Return value

On success returns the size (an unsigned integral value) of a linked list.

Failure results in undefined behavior

Example

Output:

<pre>1 #include <stdio.h> 2 #include "slltool.h" 3 4 int main () 5 { 6 Node *head; 7 int num = 2; 8 9 createList(&head); 10 11 insertElement(&head, 1, &num, sizeof(int)); 12 insertElement(&head, 2, &num, sizeof(int)); 13 insertElement(&head, 3, &num, sizeof(int)); 14 15 printf("List size: %d\n", getListSize(&head)); 16 17 return 0; 18 }</pre>	<pre>List size: 3</pre>
--	-------------------------

[Back to top](#)

getElementValue

```
void* getElementValue(Node **head, int index);
```

Returns the address of a singly linked list's value.

Returns a pointer to a linked list's data pointer. The returned address points to *void*. Hence, the pointer needs to be casted with an appropriate data type and dereferenced to get the value out. The function checks if the `index` is less than 1.

Function does not check if the head variable has been initialized as a linked list. Responsibility to check this is transferred to the user.

Parameters

`head` Address of a pointer which holds the address to the first element of a linked list.

`index` Position of a value in the linked list

Return value

On success returns a *void* pointer to a value in a linked list.

On failure (if `index` is less than 1) returns *NULL*.

Example

Output:

<pre>1 #include <stdio.h> 2 #include "slltool.h" 3 4 int main () 5 { 6 Node *head; 7 int num = 10, jim = 9; 8 int k; 9 10 createList(&head); 11 12 insertElement(&head, 1, &num, sizeof(int)); 13 insertElement(&head, 2, &num, sizeof(int)); 14 15 k = *((int *)getElementValue(&head, 1)); 16 printf("Element value at index %d:\n", 1); 17 printf("%d\n", k); 18 19 return 0; 20 }</pre>	<pre>Element value at index 1: 10</pre>
--	---

[Back to top](#)

print[data-type]

```
void print[...] (void *n);
```

Helps to execute the printList function

This is a list of functions which are used to cast and dereference the void pointer from the printList() function and print out a corresponding data type. The function names correlate to the data types. The ellipsis [...] represents the rest of the function name:

ShortInt	print short int	Char	print char
UnShort	print unsigned short int	UnChar	print unsigned char
UnInt	print unsigned int	Float	print float
Int	print int	Double	print double
Long	print long int	LongDouble	print long double
UnLong	print unsigned long int	String	print string
LongLong	print long long int		

Parameters

n A value which is to be printed

Return value

none.

Example

Output:

<pre>1 #include <stdio.h> 2 #include "slltool.h" 3 4 int main () 5 { 6 Node *head; 7 int num = 2; 8 9 createList(&head); 10 11 insertElement(&head, 1, &num, sizeof(int)); 12 13 //printInt will cast, dereference and print 14 //the data type as INT 15 printList(&head, printInt); 16 17 return 0; 18 }</pre>	<pre>2</pre>
--	--------------

[Back to top](#)