“slluti.h” Library’s Documentation

Description

“slluti” stands for “Singly Linked List Utility”. The library consists of several functions which are needed to implement a basic singly linked list (from now – linked list) and to access its information. The library follows the principles of a “generic linked list” and thus can work with any primary data type prescribed to the data pointer in the linked list, however, not all data types can be proccessed (see – printList, getElementValue).

The library does not account for error handling (except memory allocation) or any improper uses of its functions. It is the user’s responsibility to use the functions correctly and if not – there will be no apparent error’s from the library that functions are misused. The library is made with the intention to use it in accordance to this documentation and this documentation only.

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

Library uses standard libraries: <stdio.h>, <stdlib.h>

Structures:

Node	A struct to create the 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 an element from a linked list
getListSize	Returns the size of a linked list
getElementValue	Returns the address of a linked list value

Hidden from the user:

print[data-type]	Helps to execute the printList function

Node

Singly linked list structure

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

The structure contains 2 members which are:

Member	Type	Meaning
`data`	`void*`	stores the value at the 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 a list was initialized beforehand on the variable. Responsibility to check this is transferred 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

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  //Pointer to hold the address of the first linked list element
  Node *first;

  //Initialize the first element and create a linked list
  createList(&first);

  return 0;
}
```

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*. 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` An address of a pointer which holds the address to the first element of a linked list.

Return value

none.

Example

Output:

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  Node *head;
  int num = 2;

  createList(&head);

  insertElement(&head, 1, &num, sizeof(int));
  insertElement(&head, 2, &num, sizeof(int));

  printf("Size before: %d\n", getListSize(&head));
  deleteList(&head);
  printf("Size after: %d\n", getListSize(&head));

  return 0;
}
```

```
Size before: 2
Size after: 0
```

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). This 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 transferred to the user.

Parameters

`head`	An address of a pointer which holds the address to the first element of a linked list.
`callPrint`	A function name from the print[data-type] function list to print the data type which was included in the linked list.

Return value

none.

Example

Output:

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  Node *head;
  int num = 2;

  createList(&head);

  insertElement(&head, 1, &num, sizeof(int));
  insertElement(&head, 2, &num, sizeof(int));
  insertElement(&head, 3, &num, sizeof(int));

  printf("Elements of a linked list:\n");
  printList(&head, printInt);

  return 0;
}
```

```
Elements of a linked list:
2 2 2
```

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. The function checks if the linked list index is less than 1.

Function does not check if the head variable has been initialized as a linked list. Also, function does not check if index is greater than list size minus 1. Responsibility to check this is transferred to the user.

Parameters

`head`	An 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:

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  Node *head;
  int num = 1, jim = 2;

  createList(&head);

  insertElement(&head, 1, &num, sizeof(int));
  printf("First insert:\n");
  printList(&head, printInt);
  insertElement(&head, 1, &jim, sizeof(int));
  printf("Second insert:\n");
  printList(&head, printInt);

  return 0;
}
```

```
First insert:
1
Second insert:
2 1
```

Back to top

deleteElement

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

Deletes a node from the 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 the linked list index is less than 1.

Function does not check if the head variable has been initialized as a linked list. Also, function does not check if index is greater than list size minus 1. Responsibility to check this is transferred to the user.

Parameters

`head`	An 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:

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  Node *head;
  int num = 1, jim = 2;

  createList(&head);

  insertElement(&head, 1, &num, sizeof(int));
  insertElement(&head, 2, &jim, sizeof(int));
  printf("List before:\n");
  printList(&head, printInt);
  deleteElement(&head, 2);
  printf("List after: \n");
  printList(&head, printInt);

  return 0;
}
```

```
List before:
1 2
List after:
1
```

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.

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` An 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.

On failure returns *-1*.

Example

Output:

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  Node *head;
  int num = 2;

  createList(&head);

  insertElement(&head, 1, &num, sizeof(int));
  insertElement(&head, 2, &num, sizeof(int));
  insertElement(&head, 3, &num, sizeof(int));

  printf("List size: %d\n", getListSize(&head));

  return 0;
}
```

```
List size: 3
```

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 lists' value. 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 linked list index is less than 1.

Function does not check if the head variable has been initialized as a linked list. Also, function does not check if index is greater than list size minus 1. Responsibility to check this is transferred to the user.

Parameters

`head` An 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 returns *NULL*.

Example

Output:

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  Node *head;
  int num = 10, jim = 9;
  int k;

  createList(&head);

  insertElement(&head, 1, &num, sizeof(int));
  insertElement(&head, 2, &num, sizeof(int));

  k = *((int *)getElementValue(&head, 1));
  printf("Element value at index %d:\n", 1);
  printf("%d\n", k);

  return 0;
}
```

```
Element value at index 1:
10
```

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 the void pointer taken from the printList() function and print out the corresponding primary data types. The function names correlate to the data types. The ellipsis `[...]` represents the rest of the function names:

`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 pointer to a value which is to be inserted.

Return value

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

On failure returns *-1*.

Example Output:

```
#include <stdio.h>
#include "slluti.h"

int main ()
{
  Node *head;
  int num = 2;

  createList(&head);

  insertElement(&head, 1, &num, sizeof(int));

  //printInt will cast and print the data type
  printList(&head, printInt);

  return 0;
}
```

```
2
```

Back to top