



Sample Solution to Assignment 2, Problem 1

« [Back to Assignments](#)

COURSE HOME

```
#include <"list.h">
```

SYLLABUS

```
#include <stdio.h>
#include <stdlib.h>
```

CALENDAR

```
struct List_node_s {
    List_node *next;
    int value;
};
```

GETTING STARTED

LECTURE NOTES

```
List empty_list( void ) {
    return (list) { .length = 0, .front = NULL };
}
```

ASSIGNMENTS



```
List_node* create_node( int value ) {
    List_node *new_node = malloc( sizeof( List_node ) );
    new_node->value = value;
    new_node->next = NULL;
    return new_node;
}
```

RELATED RESOURCES

DOWNLOAD COURSE MATERIALS

```
void list_append( List *list, int value ) {
    if( list->front == NULL ) {
        list->front = create_node( value );
    } else {
        List_node *p = list->front;
        for( size_t i = 1; i < list->length; ++i, p = p->next );
    }
}
```

```

        p->next = create_node( value );
    }
    ++list->length;
}

void list_delete_from_front( List *list, int value ) {
    List_node *front = list->front;
    while( front != NULL && front->value == value ) {
        list->front = front->next;
        --list->length;
        free( front );
        front = list->front;
    }
}

void list_delete( List *list, int value ) {
    list_delete_from_front( list, value );
    if( list->front == NULL ) {
        return;
    }

    List_node *prev = list->front;
    List_node *p = list->front->next;

    while( p != NULL ) {
        if( p->value == value ) {
            prev->next = p->next;
            free( p ); --list->length;
            p = prev->next;
        } else {
            prev = p;
            p = prev->next;
        }
    }
}

void list_insert_before( List *list, int insert, int before ) {
    if( list->front != NULL && list->front->value == before ) {
        List_node *new_node = create_node( insert );
        new_node->next = list->front;
        list->front = new_node;
        ++list->length;
    } else {

```

```

List_node *prev = list->front;
List_node *next = list->front->next;
while( next != NULL ) {
    if( next->value == before ) {
        prev->next = create_node( insert );
        prev->next->next = next;
        ++list->length; return;
    }
    prev = next;
    next = next->next;
}
}
}

```

```

void list_apply( List *list, int (*function_ptr)( int) ) {
    for( List_node *p = list->front; p != NULL; p = p->next ) {
        p->value = (*function_ptr)( p->value );
    }
}

```

```

int list_reduce( List *list, int (*function_ptr)(int, int) ) {
    if( list->front == NULL ) {
        return 0;
    }

```

```

    int result = list->front->value;

```

```

    for( List_node *p = list->front->next; p != NULL; p = p->next ) {
        result = (*function_ptr)( result, p->value );
    }

```

```

    return result;
}

```

```

void list_print( List list ) {
    if( list.front == NULL ) {
        printf( "{}\n" );
    } else {
        printf( "{ " );

```

```

        List_node *p = list.front;
        size_t length = list.length;

```

```

while( p->next != NULL && length > 0 ) {
    printf( "%d -> ", p->value );
    p = p->next; --length;
}
printf( "%d }\n", p->value );

if( length != 1 ) {
    printf( "Error: badly formed list.\n" );
    exit( EXIT_FAILURE );
}
}
}

void list_clear( List *list ) {
    List_node *front = list->front;
    size_t length = list->length;

    while( front != NULL && length > 0 ) {
        List_node *next = front->next;
        free( front );
        front = next;
        --length;
    }

    if( length != 0 ) {
        printf( "Error: failed to clean up list properly.\n" );
        exit( EXIT_FAILURE );
    }
}

```

Below is the output using the test data:

list:

```

1: OK [0.002 seconds] OK!
2: OK [0.004 seconds] OK!
3: OK [0.035 seconds] OK!
4: OK [2.175 seconds] OK!
5: OK [0.133 seconds] OK!
6: OK [0.305 seconds] OK!
7: OK [0.061 seconds] OK!
8: OK [0.213 seconds] OK!

```

9: OK [0.002 seconds] OK!

10: OK [1.054 seconds] OK!

« [Back to Assignments](#)

FIND COURSES

- » Find by Topic
- » Find by Course Number
- » Find by Department
- » New Courses
- » Most Visited Courses
- » OCW Scholar Courses
- » Audio/Video Courses
- » Online Textbooks
- » Instructor Insights
- » Supplemental Resources
- » MITx & Related OCW Courses
- » MIT Open Learning Library
- » Translated Courses

FOR EDUCATORS

- » Chalk Radio Podcast
- » OCW Educator Portal
- » Instructor Insights by Department
- » Residential Digital Innovations
- » OCW Highlights for High School
- » Additional Resources

GIVE NOW

- » Make a Donation
- » Why Give?
- » Our Supporters
- » Other Ways to Contribute
- » Become a Corporate Sponsor

ABOUT

- » About OpenCourseWare
- » Site Statistics
- » OCW Stories
- » News
- » Press Releases

TOOLS

- » Help & FAQs
- » Contact Us
- » Site Map
- » Privacy & Terms of Use
- » RSS Feeds

OUR CORPORATE SUPPORTERS



ABOUT MIT OPENCOURSEWARE

MIT OpenCourseWare makes the materials used in the teaching of almost all of MIT's subjects available on the Web, free of charge. With more than 2,400 courses available, OCW is delivering on the promise of open sharing of knowledge. [Learn more »](#)



© 2001–2020
Massachusetts Institute of Technology



Your use of the MIT OpenCourseWare site and materials is subject to our [Creative Commons License](#) and other [terms of use](#).