

## Lab #6

### Summer 2024

### Requirements

In this lab, you will cover creating and maintaining linked lists. Your list implementation should make use of **one placeholder (or "dummy") node at the beginning of the list**. Remember that placeholder nodes are not considered an element in the list, and must be invisible to the user of your list. In this lab you are given the following struct definition:

```
struct Node {  
    Node * next;  
    void *data;  
};
```

#### 1.1 makeList

```
int makeList(Node **list)
```

❶

**Info:** This function will initialize the provided pointer to Node \* to an empty list. It will return 0 if initialization was successful, or 1 if it was not.

#### 1.2 getSize

```
int getSize(Node *list)
```

❶

**Info:** This function takes a list, and returns the number of elements on the list. Note that if the list is empty, the number of elements on the list is 0.

#### 1.3 getAtIndex

```
void * getAtIndex(Node *list, int index)
```

❶

**Info:** This function takes a list, and returns the data stored at the given index, or **NULL** if the index is invalid. **Note that an invalid index is one which is < 0 or >= getSize(list).**

#### 1.4 insertAtTail

```
int insertAtTail(Node *list, void *data)
```

❶

**Info:** This function takes a pointer to Node, and inserts the given data at the tail of the list. It returns 0 if insertion was successful, or 1 if insertion failed.

#### 1.5 removeFromHead

```
void * removeFromHead(Node *list)
```

❶

**Info:** This function takes a pointer to Node, and removes the element at the head of the list (if any). It will return the data which was removed from the list, or NULL if the list was empty. You may assume that no data on the list will be NULL.

#### 1.6 freeList

```
void freeList(Node **list)
```

❶

**Info:** This function takes a pointer to Node \*, and frees the memory allocated to the list. After freeing, it sets the pointer to NULL. **Note that the data on the list is not considered part of the memory allocated to the list.**

## Submission Information

Submit this assignment by uploading your lab6.c file using the mucsmake command.

Use the following command on Hellbender:

```
mucsmake <course> <assignment> <filename>
```

For example:

```
mucsmake 2050 lab6 lab6.c
```

## Rubric: 21 points

1. Write required *makeList* function
  - \* 2 points
2. Write required *getSize* function
  - \* 3 points
3. Write required *getAtIndex* function
  - \* 3 points
4. Write required *insertAtTail* function
  - \* 5 points
5. Write required *removeFromHead* function
  - \* 5 points
6. Write required *freeList* function
  - \* 3 points

## Notice:

1. All of your lab submissions **must** include documentation in the form of code comments to receive full points. In addition, your program is expected to have a **comment header** at the top that includes your name, pawprint, the course you are taking, and the lab that you solved. You can refer to the Lab 0 document for an example of the comment header.
2. All of your lab submissions must compile under GCC using the -Wall and -Werror flags to be considered for a grade. These flags will automatically be applied if you use the compile command.
3. Do **NOT** change the given function prototype or anything else in the provided .h file.