```c
#include <cs50.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct node
{
    string phrase;
    struct node *next;
} node;

#define LIST_SIZE 2

bool unload(node *list);
void visualizer(node *list);

int main(void)
{
    node *list = NULL;

    // Add items to list
    for (int i = 0; i < LIST_SIZE; i++)
    {
        string phrase = get_string("Enter a new phrase: ");

        // TODO: add phrase to new node in list

        // Visualize list after adding a node.
        visualizer(list);
    }

    // Free all memory used
    if (!unload(list))
    {
        printf("Error freeing the list.\n");
        return 1;
    }

    printf("Freed the list.\n");
    return 0;
}

```

```c
bool unload(node *list)
{
    // TODO: Free all allocated nodes
    return false;
}

void visualizer(node *list)
{
    printf("\n+-- List Visualizer --+\n\n");
    while (list != NULL)
    {
        printf("Location %p\nPhrase: \"%s\"\nNext: %p\n\n", list, list->phrase, list->next);
        list = list->next;
    }
    printf("+---------------------+\n\n");
}
```

```c
#include <cs50.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct node
{
    string phrase;
    struct node *next;
} node;

node *table[26];

int hash(string phrase);
bool unload(node *list);
void visualizer(node *list);

int main(void)
{
    // Add items
    for (int i = 0; i < 3; i++)
    {
        string phrase = get_string("Enter a new phrase: ");

        // Find phrase bucket
        int bucket = hash(phrase);
        printf("%s hashes to %i\n", phrase, bucket);
    }
}

// TODO: return the correct bucket for a given phrase
int hash(string phrase)
{
    return 0;
}
```