**Question:**

You are given a singly linked list class. This class has two head nodes, such that it can contain two separate linked lists. Your task is to complete the appropriate function that, on satisfying a given condition, takes the linked list pointed to by the first head pointer, and splits it into two Linked Lists, such that the second head pointer becomes the head pointer of the second list.

**Starer Code and Submission:**

You must complete the following two functions:

1. `int split(string searchKey){...}` **(35 points)**
2. `~SLL(){...}` **(5 points)**

Starter code with three files will be provided:

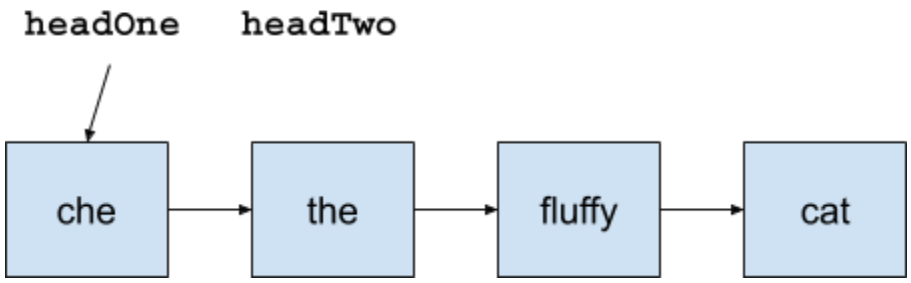
1. SLL.hpp - header file
2. SLL.cpp - definitions file
3. driver.cpp - driver file

You are to download the starter code and complete the missing function in the definitions file. You are not allowed to change any of the existing member functions or add new data members. You can feel free to add additional test cases to the driver file. Note that you are not guaranteed to receive full points just by passing the given test cases. Your algorithm also has to be correct and not introduce any memory leaks. **When ready to submit, zip all the completed files together and upload to the Midterm 1 Coding Submission link.**

**Specifications:**

1. The `split` function takes in a `searchKey` value.
2. If `searchKey` exists, the linked list gets split into two separate lists:
    a. The node matching `searchKey` becomes the `headTwo`
    b. If the searchKey is found at `headOne,` then the first list becomes an empty list (`headOne` gets set to `nullptr),` and `headTwo` becomes the head of the list
    c. Function returns integer value of 0
3. If `searchKey` does not exist in the linked list, the function does nothing and returns an integer value of 1.
4. If both the Linked List(s) are empty, return an integer value of 1.

5. If **`headTwo`** is already pointing to a node, the function does nothing and returns an integer value of 2
6. Assume a constructor is defined and initializes the both the head nodes to **`nullptr.`** It is also overloaded such that if it is called with the string “demo” it will generate a single linked list with the strings “che”,”the”, “fluffy”, “cat”, with headOne pointing to the node containing “che”, and headTwo pointing to **`nullptr`**. See example below.
7. Complete the destructor definition such that all of the dynamic memory is properly deallocated.




The class struct and class definitions are given as follows:

```
struct Node{
     string key;
     Node *next;
};

class SLL{
private:
     Node* headOne;
     Node* headTwo;

public:
     // TO DO: Implement the definition for these prototype:
     SLL();
     SLL(string demo);
     ~SLL();
```

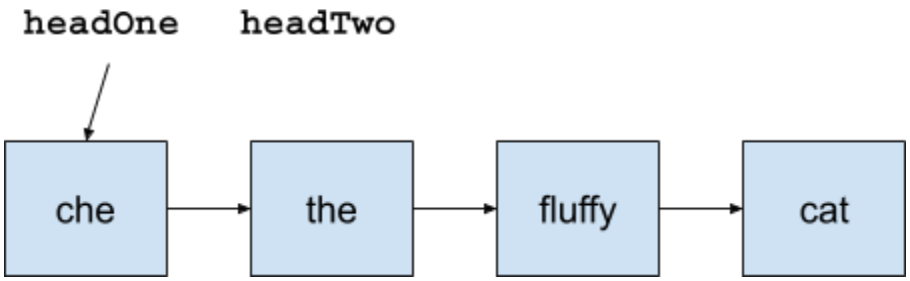
```
    int split(string searchKey);
    //Other prototype definitions can be found in the starter
    // code.
};
```

**See example on the next page.**

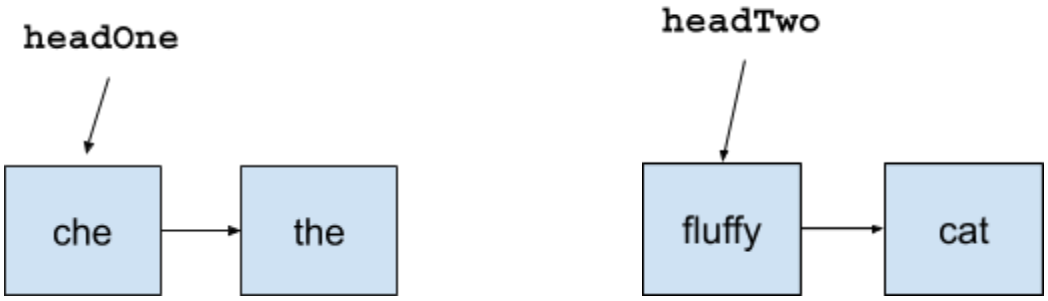
**Example:**
*in main:*

```
SLL A("demo");
int result = A.split("fluffy");
```

**Before:**




**After:**




`result` holds is set to 0.