Find the first non-repeating character from a stream of characters

Given a stream of characters, find the first non-repeating character from stream. You need to tell the first non-repeating character in O(1) time at any moment.

If we follow the first approach discussed here, then we need to store the stream so that we can traverse it one more time to find the first non-repeating character at any moment. If we use extended approach discussed in the same post, we need to go through the count array every time first non-repeating element is queried. We can find the first non-repeating character from stream at any moment without traversing any array.

We strongly recommend you to minimize the browser and try it yourself first.

The idea is to use a DLL (**D**oubly **L**inked **L**ist) to efficiently get the first non-repeating character from a stream. The DLL contains all non-repeating characters in order, i.e., the head of DLL contains first non-repeating character, the second node contains the second non-repeating and so on.
We also maintain two arrays: one array is to maintain characters that are already visited two or more times, we call it repeated[], the other array is array of pointers to linked list nodes, we call it inDLL[]. The size of both arrays is equal to alphabet size which is typically 256.

```
1) Create an empty DLL. Also create two arrays inDLL[] and repeated[] of size 256.
   inDLL is an array of pointers to DLL nodes. repeated[] is a boolean array,
   repeated[x] is true if x is repeated two or more times, otherwise false.
   inDLL[x] contains pointer to a DLL node if character x is present in DLL,
   otherwise NULL.

2) Initialize all entries of inDLL[] as NULL and repeated[] as false.

3) To get the first non-repeating character, return character at head of DLL.

4) Following are steps to process a new character 'x' in stream.
  a) If repeated[x] is true, ignore this character (x is already repeated two
      or more times in the stream)
  b) If repeated[x] is false and inDLL[x] is NULL (x is seen first time)
     Append x to DLL and store address of new DLL node in inDLL[x].
  c) If repeated[x] is false and inDLL[x] is not NULL (x is seen second time)
     Get DLL node of x using inDLL[x] and remove the node. Also, mark inDLL[x]
     as NULL and repeated[x] as true.
```

Note that appending a new node to DLL is O(1) operation if we maintain tail pointer. Removing a node from DLL is also O(1). So both operations, addition of new character and finding first non-repeating character take O(1) time.

C/C++ Python

```
// A C++ program to find first non-repeating character
// from a stream of characters
#include <iostream>
#define MAX_CHAR 256
using namespace std;

// A linked list node
struct node
{
    char a;
    struct node *next, *prev;
};

// A utility function to append a character x at the end
// of DLL. Note that the function may change head and tail
// pointers, that is why pointers to these pointers are passed.
void appendNode(struct node **head_ref, struct node **tail_ref,
                char x)
{
    struct node *temp = new node;
    temp->a = x;
    temp->prev = temp->next = NULL;

    if (*head_ref == NULL)
    {
        *head_ref = *tail_ref = temp;
        return;
    }
    (*tail_ref)->next = temp;
    temp->prev = *tail_ref;
    *tail_ref = temp;
}

// A utility function to remove a node 'temp' fromt DLL.
// Note that the function may change head and tail pointers,
// that is why pointers to these pointers are passed.
void removeNode(struct node **head_ref, struct node **tail_ref,
                struct node *temp)
{
    if (*head_ref == NULL)
        return;

    if (*head_ref == temp)
        *head_ref = (*head_ref)->next;
    if (*tail_ref == temp)
        *tail_ref = (*tail_ref)->prev;
    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    if (temp->prev != NULL)
        temp->prev->next = temp->next;

    delete(temp);
}

void findFirstNonRepeating()
{
    // inDLL[x] contains pointer to a DLL node if x is present
    // in DLL. If x is not present, then inDLL[x] is NULL
    struct node *inDLL[MAX_CHAR];

    // repeated[x] is true if x is repeated two or more times.
    // If x is not seen so far or x is seen only once. then
    // repeated[x] is false
    bool repeated[MAX_CHAR];

    // Initialize the above two arrays
    struct node *head = NULL, *tail = NULL;
    for (int i = 0; i < MAX_CHAR; i++)
    {
        inDLL[i] = NULL;
        repeated[i] = false;
    }

    // Let us consider following stream and see the process
    char stream[] = "geeksforgeeksandgeeksquizfor";
    for (int i = 0; stream[i]; i++)
    {
        char x = stream[i];
        cout << "Reading " << x << " from stream \n";

        // We process this character only if it has not occurred
        // or occurred only once. repeated[x] is true if x is
        // repeated twice or more.s
        if (!repeated[x])
        {
            // If the character is not in DLL, then add this at
            // the end of DLL.
            if (inDLL[x] == NULL)
            {
                appendNode(&head, &tail, stream[i]);
                inDLL[x] = tail;
            }
            else // Otherwise remove this character from DLL
            {
                removeNode(&head, &tail, inDLL[x]);
                inDLL[x] = NULL;
                repeated[x] = true; // Also mark it as repeated
            }
        }

        // Print the current first non-repeating character from
        // stream
        if (head != NULL)
            cout << "First non-repeating character so far is "
                 << head->a << endl;
    }
}

/* Driver program to test above function */
int main()
{
    findFirstNonRepeating();
    return 0;
}
```

Run on IDE

Output:

```
Reading g from stream
First non-repeating character so far is g
Reading e from stream
First non-repeating character so far is g
Reading e from stream
First non-repeating character so far is g
Reading k from stream
First non-repeating character so far is g
Reading s from stream
First non-repeating character so far is g
Reading f from stream
First non-repeating character so far is g
Reading o from stream
First non-repeating character so far is g
Reading r from stream
First non-repeating character so far is g
Reading g from stream
First non-repeating character so far is k
Reading e from stream
First non-repeating character so far is k
Reading e from stream
First non-repeating character so far is k
Reading k from stream
First non-repeating character so far is s
Reading s from stream
First non-repeating character so far is f
Reading a from stream
First non-repeating character so far is f
Reading n from stream
First non-repeating character so far is f
Reading d from stream
First non-repeating character so far is f
Reading g from stream
First non-repeating character so far is f
Reading e from stream
First non-repeating character so far is f
Reading e from stream
First non-repeating character so far is f
Reading k from stream
First non-repeating character so far is f
Reading s from stream
First non-repeating character so far is f
Reading q from stream
First non-repeating character so far is f
Reading u from stream
First non-repeating character so far is f
Reading i from stream
First non-repeating character so far is f
Reading z from stream
First non-repeating character so far is f
Reading f from stream
First non-repeating character so far is o
Reading o from stream
First non-repeating character so far is r
Reading r from stream
First non-repeating character so far is a
```