Lab 7

October 27, 2021

This lab is primarily implementation-based. We want you to implement lists, stacks, and queues that were done in the class. Each node in these data structures will be a structure that contains two variables: the first variable represents the value at the node, and the second represents the pointer to the next node.

```c
#include<stdio.h>
struct node
{
    int value;
    struct node *next;
};
```

We can allocate a memory for a node as follows (Note that you have to include a new library stdlib.h to use the malloc function):

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int value;
    struct node *next;
};

int main()
{
  struct node *v = (struct node *)malloc(sizeof(struct node));
  return 0;
}
```

Observe that the malloc function always returns a pointer. Thus, we store the address of the newly created memory location in the pointer variable "v". We can access variables of the structure via $v->value$ and $v->next$.

Sometimes, you may want to delete a node from your data structure. To this end, you can use the following function: *free()*

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int value;
    struct node *next;
};

int main()
{
  struct node *v = (struct node *)malloc(sizeof(struct node));
  v->value = 2;
  v->next = NULL;


  free(v);
  return 0;
}
```

1. List

   Implement lists as done in the class. You will need to perform two functions:
   ADD(x): Adds an integer x at the start of the list.
   DELETE(x): Delete the integer x from the list.

   Input: The first input line contain an integer $n(1 \leq n \leq 1000)$. The next n lines will be one of the following types:

   1. $a\ x$
      Add the integer x at the front of the list.
   2. $d\ x$
      Delete an integer x from the list. If x is not present in the list, do nothing.

   Output: Print the list at the end of all the operations. The first integer to be printed is the value at the first node in the list, then the second node, and so on. If the list is empty, print "NULL".

   Input : 5
   a 1
   a 2
   d 3
   a 3
   d 1
   Output : 2 3

2. Stack

   Implement Stacks as done in the class. You will need to perform two functions:
   PUSH(x): Adds an integer a at the top of the stack.
   POP(): Remove the top element of the stack.

   Input: The first input line contain an integer $n(1 \leq n \leq 1000)$. The next n lines will be one of the following types:

   1. $a\ x$
      Add the integer x at the top of the list.
   2. d
      Remove the top element of the stack.

   Output: Print the stack after each operation on a new line. The top element of the stack should be printed first, then the second top-most element, and so on. If the stack is empty, print "NULL".

   Input : 4
   a 1
   d
   a 2
   a 3
   Output : 1
   NULL
   2
   3 2

3. Queue

   Implement Queues as done in the class. You will need to perform two functions:
   ENQUE(x): Adds an integer a at the rear of the queue.

DEQUEUE(): Remove an integer from the front of the queue.

Input: The first input line contain an integer $n(1 \leq n \leq 1000)$. The next n lines will be one of the following types:

1. $e\ x$
   Add the integer x at the rear end of the queue.
2. d
   Remove an integer from the front of the queue.

Output: Print the queue after each operation on a new line. The front element of the queue should be printed first, then the second element, and so on. If the queue is empty, print "NULL".

Input	:	4
		e 1
		d
		e 2
		e 3
Output	:	1
		NULL
		2
		2 3