EE 2331 Data Structures and Algorithms, Semester B, 2009/10

Tutorial 5: Radix Conversion

Week 5 (11th February, 2010)

The tasks of tutorial exercises are divided into three levels. Level 1 is the basic tasks. You should have enough knowledge to complete them after attending the lecture. Level 2 is the advanced tasks. You should able to tackle them after revision. Level 3 is the challenge tasks which may be out of the syllabus and is optional to answer. I expect you to complete at least task A in the tutorial.

Outcomes of this tutorial

1. Able to implement stacks using doubly linked list
2. Able to operate stacks
3. Able to use stacks to perform radix conversion

Stacks are a data structure that exhibits the Last-In First-Out (LIFO) characteristic, which is useful in a number of applications. In this tutorial, we will use a doubly linked list with dummy header to implement a stack.

A stack implemented by doubly and linear linked list with dummy header can be visualized as below. The dummy header always points to the top element if it exists, or points to NULL otherwise.

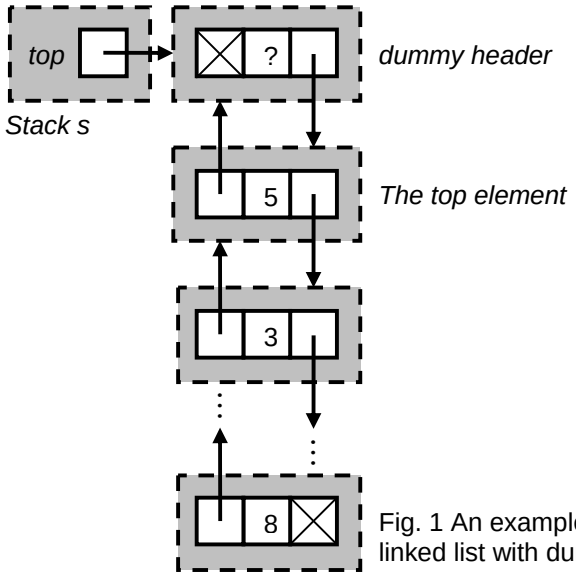

Fig. 1 An example stack implemented by doubly linked list with dummy header pointed by *top*.

The basic stack operations like *initialize*, *destroy*, and *check empty* have been provided to you except *push* and *pop*.

Task A (Level 2): Push

Implement the C function **void push(Stack *s, int e)** that accepts a pointer *s* to a stack and an integer *e* and inserts a node containing *e* to the top of the stack.
For example, *s* is the stack in figure 1:

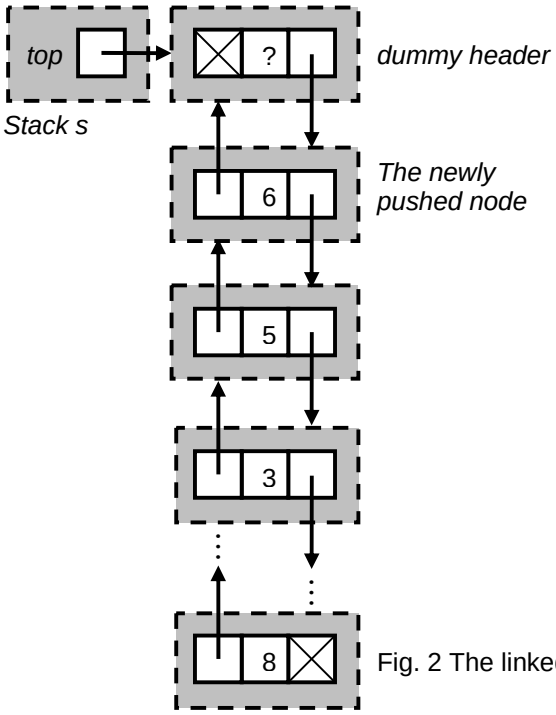

Fig. 2 The linked list after calling push(&*s*, 6).

Discussion: Why do we design the push functions to accept a **pointer to stack structure** (i.e. Stack **s*) instead of accepting the **stack structure** (i.e. Stack *s*) instead?

Task B (Level 2): Pop

Write another function, **int pop(Stack *s)**, that deletes the top node from *s* if *s* is non-empty and returns the value of the top node to the calling function. For example, *s* is the stack in figure 1:

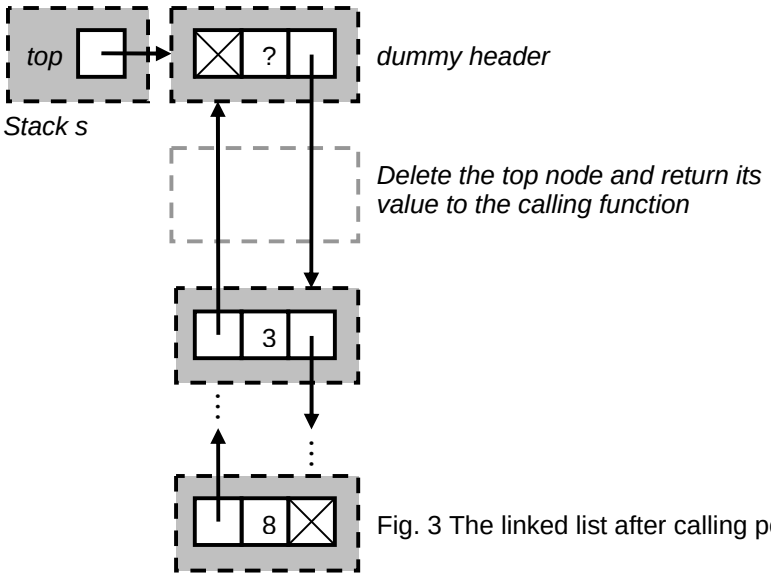

Fig. 3 The linked list after calling pop(&s).

Discussion: What will happen if you forget to "*free*" the deleted node?

Task C (Level 1): Radix Conversion Using Stack

Write another C function **int convert(int *a*, int *r*)** that accepts two integers *a* and *r*, convert number *a* from decimal to radix *r* and **returns the converted number**. *Note: you may assume a is a positive number, and r is greater than 1 and smaller than 10*

To convert a number from decimal to radix *r*, divide the number *a* by *r* and a remainder is produced. Push the remainder into the stack. Repeatedly divide the quotient and push the remainder into the stack until the quotient becomes zero. Finally pop out the digits from the stack and that will be the converted number.

Expected Output:

```
Enter the number a: 6
Enter the new radix r: 2
Answer: 110
```

```
Enter the number a: 23
Enter the new radix r: 8
Answer: 27
```