Lab 4: Stack and its Applications

Objectives

- To understand the concept of stack data structures and their implementation using singly linked list.
- To understand the basic operations related to stack and queue.
- To write simple applications that use stack and queue and apply more complex operations.

Review

A stack is a special type of SList, where the elements are accessed, inserted, and removed only from one end.

Basic Operations

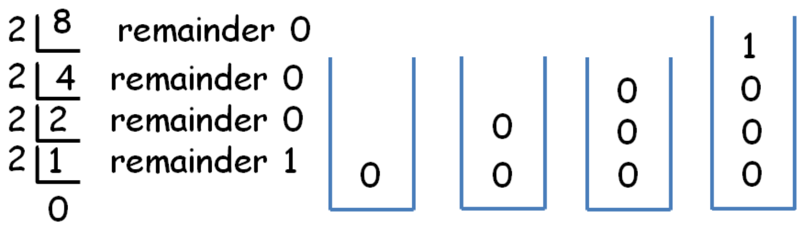
- o Push(T item) adds an item to the top of the stack.
- o Pop() removes the top element from the stack and returns the removed element.
- o Peek() accesses the top node of a stack and returns its element.

Stack applications

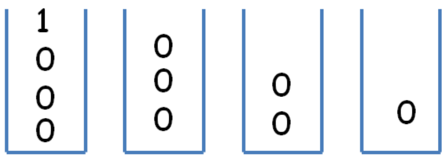
1. Convert Decimal to Binary

Conversion from a decimal number to a binary number can be done easily with stack using the following procedures.

Keep the remainder of the input divided by 2 in a stack, let the input become the integer value of the input divided by 2. Repeat this process until the input is 0.



To get the binary value, repeatedly pop and printout the popped value.



print 1 0 0 0

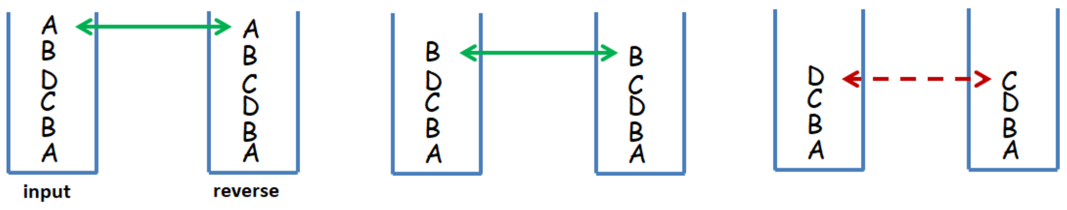
2. Palindrome

A palindrome is a word, phase, number or other sequence of units that can be read the same way in either direction (the adjustment of punctuation and spaces between words is generally permitted). Some examples of palindrome are civic, radar, level, rotor, kayak, reviver, racecar, redder, madam.

Stack can be utilized to solve this problem. Assume a sequence of letters or numbers that we would like to check is in the stack. Here are the procedures.

1. Create another stack keeps the content of the input stack in the backward order. Make sure the input stack remain unchanged.
2. One by one compare the top element of the stack. If they are not the same, the sequence isn't a palindrome and quit the program. If they are the same, pop them out from both stacks and repeat this step until both stack are empty.

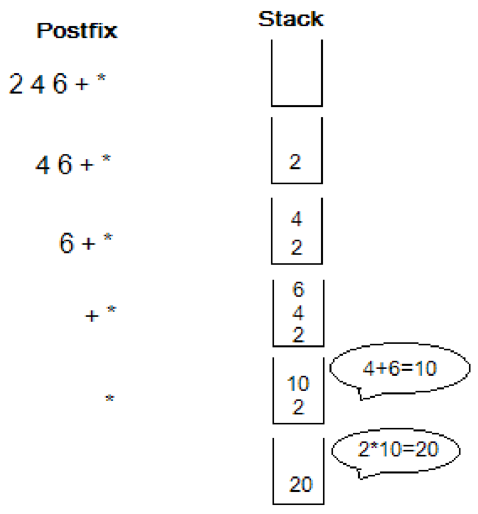
An example of an input sequence ABDCBA and a few steps that show that the input is not a palindrome.



3. **Evaluating Postfix Expressions**

   Postfix expression is an expression that an operator follows its operands ex. 3 4 +, and 12 3 / 5 4 - 1 9 + + -

   This can be solved using a stack in the following steps. An element is read from a postfix expression. If the read element is an operand, it is pushed to a stack. If it is an operator (opt), it pops two more elements (y and x, respectively) from a stack. It evaluates x and y with the opt operator, ie. x opt y, then it pushes the result to the stack. These steps are repeated until all elements in the postfix expression are read and the stack is empty.



The Stack, SList and Node UMLs are as follows.

```
SList<T>
---------------------------------------------
size: int
first: Node<T>
last: Node<T>
---------------------------------------------
SList()
addFirst(element: T): void
addLast(element: T): void
addAtIndex(index: int, element: T): void
removeFirst(): T
removeLast(): T
removeAtIndex(index: int): T
isEmpty(): Boolean
getSize(): int
printHorizontal(): void
```

```
Node<T>
---------------------------------------------
element: T
next: Node<T>
---------------------------------------------
Node(T element)
```

```
Stack<T>
---------------------------------------------
- list : SList<T>
---------------------------------------------
Stack()
push(element : T) : void
pop() : T
peek() : T
printVertical() : void
...
```

<u>Exercise 1</u>

a) Implement push(element). Test the result in the main() method. Use the given template source code to complete this exercise.
b) Implement pop(). Test the result in the main() method. Use the given template source code to complete this exercise.
c) Implement peek(). Test the result in the main() method.

Exercise 2 Implement binaryConversion(). This method converts a decimal number in the input parameter to a binary number and prints its result. Test the result in the main() method. Use the given template source code to complete this exercise.

Exercise 3 Implement a method reverseStack that returns another stack containing elements in the reverse order. **The method should preserve the content of the input stack. i.e. the input stack should not be empty** after the end of the method. Note that there is no pseudocode or figure for this one.

Exercise 4 Implement boolean isPalindrome(String word). The input parameter of this method is a string that you want to test if it is a palindrome. Test the result in the main() method. Use the given template source code to complete this exercise. Hint: Use char charAt(int index) method to extract the character at the specific index from the string. You may utilize the method reverseStack you implemented in Exercise 3.

Exercise 5 Implement evalPostfix(String[] input). In the template there is an isInteger() method provided for check the input is integer or not. You can use it to help in the implementation. Test the result in the main() method. Use the given template source code to complete this exercise.