Stack

The stack is a LIFO (Last-in First-out) data structure

The only data element that can be removed is the most recently added element

Stack ADT

Specification

Elements: Stack elements can be of any type, but we will assume StackElement

Structure: Any mechanism for determining the elements order of arrival into the stack

Stack ADT Continued

Domain: The number of stack elements is bounded. A stack is considered full if the upper-bound is reached. A stack with no elements is considered empty.

type Stack;

Operations: There are seven operations as follows:

Stack ADT Continued

function create (s: Stack, isCreated: boolean)
results: if s cannot be created, isCreated is false; otherwise, isCreated is true,the stack is created and is empty

function terminate (s: Stack)
results: stack s no longer exists

Stack ADT Continued

function isFull (s: Stack)
results: returns true if the stack is full; otherwise false is returned

function isEmpty (s: Stack)
results: returns true if the stack is empty; otherwise, false is returned

function push (s: Stack, e: StackElement)
requires: isFull (s) is false
results: element e is added to the stack as the most recent element

Stack ADT Continued

function pop (s: Stack, e: StackElement)
requires: isEmpty(s) is not false
results: The most recently added element is removed and assigned to e

function peek (s: Stack, e: StackElement)
requires: isEmpty(s) is not false
results: The most recently added element is assigned to e but not removed

Stack Implementation

Problem: Write stack.h.

Problem: After we agree on stack.h, write create, terminate, isFull, isEmpty, push, and pop.

Problem: A datafile words.txt contains zero or more words, one word per line. Output the word followed by “Palindrome” or “Not Palindrome”