Palindrome: A palindrome is a string that reads the same whether read forwards or backwards. The same sequence of characters is produced when reading a palindrome left-to-right or right-to-left.

Stack: A stack is an Abstract Data Type that stores objects in Last-In-First-Out (LIFO) order. The last object put onto a stack is the first object retrieved from a stack.

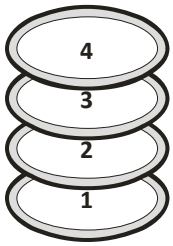

Figure 1. Stack of Plates

Example: Plates dispensed in a cafeteria.
1. First, plate 1 is put on the stack.
2. Then, plate 2 is placed on top of the stack. Access to plate 1 is not possible without first removing plate 2. Only the plate on top of the stack is accessible.
3. Put plates 3 and 4 on the stack. Now only plate 4 can be retrieved.

The last plate on the stack is the first plate to be retrieved. The last plate in the stack is the first plate out of the stack.

Note that the plates are in ***reverse*** order. If I wanted to determine that an identifier was a palindrome, I might want to put the characters of the identifier in ***reverse*** order.

Insight: A stack can be used to reverse the order of the characters in an identifier.

Palindrome checker algorithm: draft 1
1. Read an identifier. The identifier is a candidate to be a palindrome.
2. Store the characters of the candidate on a stack.
3. Extract the characters of the original candidate that was read in step one. Extract the characters, one at a time, reading from left to right.
4. Compare the characters extracted in step 3 with characters popped from the stack.
5. If all the characters match then the candidate is a palindrome.
6. If one or more characters do not match then the candidate is not a palindrome.
7. Stop when the stack is empty and the last character is extracted from the original candidate-identifier.

Example 1: Candidate is a palindrome
1. Read an identifier. The identifier is a candidate to be a palindrome. Peruse the candidate-identifier "mom" in the figure below.

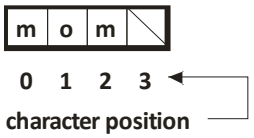

Figure 2. Candidate-identifier “mom”

2. Store the characters of the candidate on a stack.

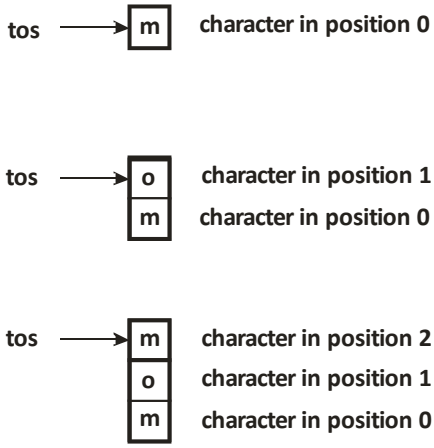

Figure 3. Pushing the characters of "mom" on the stack

3. Extract the characters of the original candidate that was read in step one. Extract the characters, one at a time, reading from left to right.
4. Compare the characters extracted in step 3 with characters popped from the stack.
5. If all the characters match then the candidate is a palindrome.
6. If one or more characters do not match then the candidate is not a palindrome.
7. Stop when the stack is empty and the last character is extracted from the original candidate-identifier.

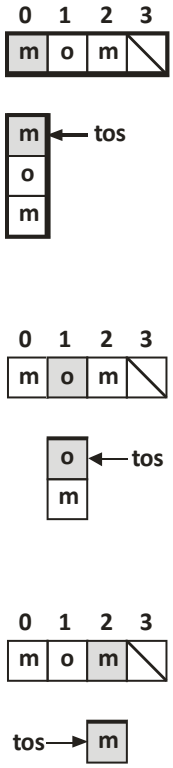

Figure 4. Comparing the characters of the string "mom" with the characters of "mom" on the stack

Example 1: Candidate is not a palindrome

1. Read an identifier. The identifier is a candidate to be a palindrome. Peruse the candidate-identifier "tilt" in the figure below.

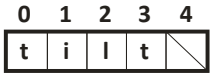

Figure 5. Candidate-identifier "tilt"

2. Store the characters of the candidate on a stack.

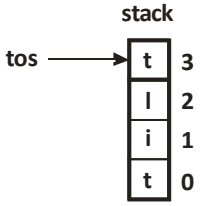

Figure 6. Candidate-identifier "tilt" on a stack

3. Extract the characters of the original candidate that was read in step one. Extract the characters, one at a time, reading from left to right.
4. Compare the characters extracted in step 3 with characters popped from the stack.
5. If all the characters match then the candidate is a palindrome.
6. If one or more characters do not match then the candidate is not a palindrome.

7. Stop when the stack is empty and the last character is extracted from the original candidate-identifier.

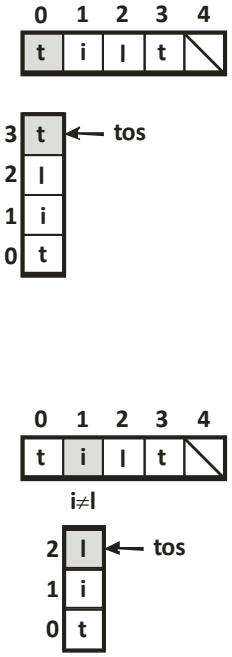

Figure 6. Matching the characters in the candidate with the characters in the stack

Create a program **p02** that processes command line arguments

```
include <iostream>
include <fstream>
int main
{    process command line arguments
     open stream ifs;
     open stream ofs;
     close stream ifs;
     close stream ofs;
     return 0;
}
```

1. Enhance program **p02** to read the identifiers in stream ifs.

```
include <iostream.h>
include <fstream.h>
void palindromemgr
     (istream& ifs
     ,ostream& ofs
     )
{    read an identifier
     while the read operation was successful do
     {    print the identifier
          read an identifier
     }
}
int main
{    process command line arguments
     open stream ifs;
     open stream ofs;
     palindromemgr(ifs,ofs);
     close stream ifs;
     close stream ofs;
     return 0;
}
```

2. Add logic to determine if the identifier is a palindrome.

```
include <stdio.h>
bool ispalindrome(char* candidate)
{    return 1;
}
void palindromemgr
     (istream& ifs
     ,ostream& ofs
     )
```

```
{   do forever
        read a candidate
        test: Is the input file stream at end of file, exit the loop
        print a new line
        print the candidate
        print " is"
        print " not" if the candidate is not a palindrome
        print " a palindrome."
    end do

    }
}
int main
{   process command line arguments
    open stream ifs;
    open stream ofs;
    palindromemgr(ifs,ofs);
    close stream ifs;
    close stream ofs;
    return 0;
}
```

3. Find out what a palindrome is.

   Palindrome: A palindrome is a string that reads the same whether read forwards or backwards. The same sequence of characters is produced when reading a palindrome left-to-right or right-to-left.

   Examples:
   3.1. mom is a palindrome
   3.2. dad is a palindrome
   3.3. totot is a palindrome
   3.4. tom is not a palindrome
   3.5. priscilla is not a palindrome

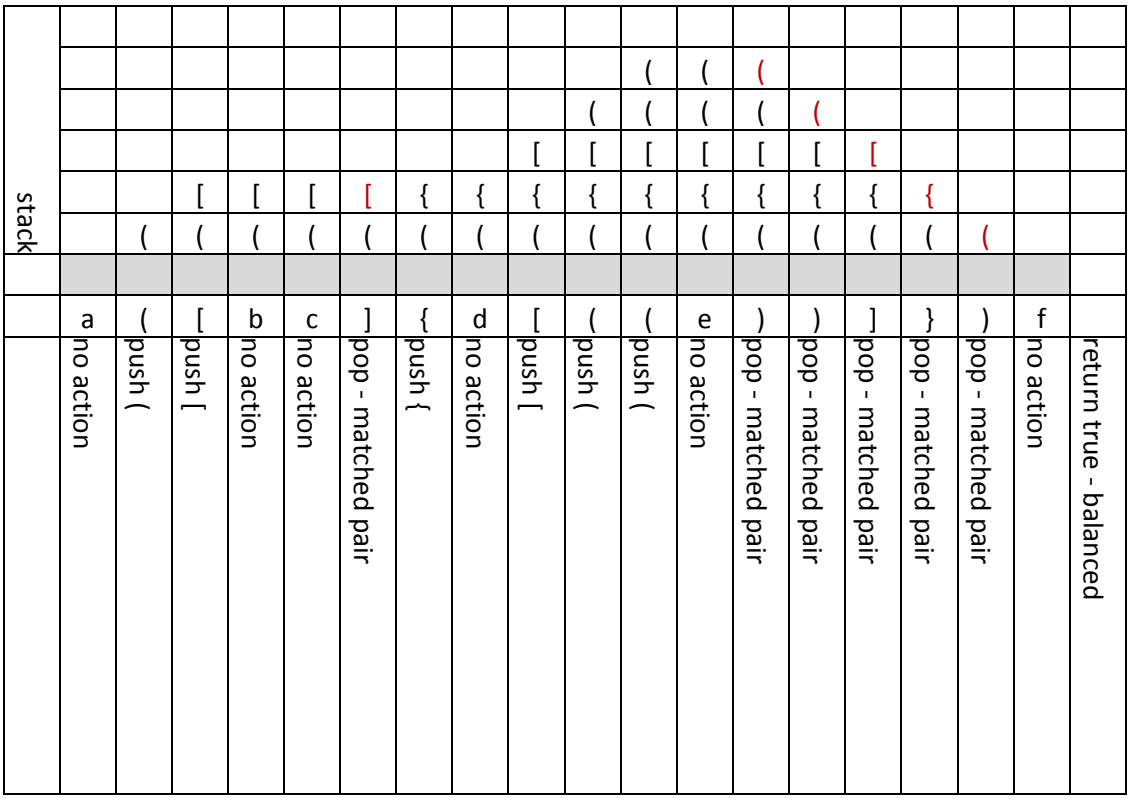

Figure 1. Algorithm for Finding Balanced Brackets Balanced

stack						
			[	[		
		(	(	(		
	a	(	[	)	b	]
	no action	push (	push [	pop - unmatched pair -> not balanced		

Figure 2. Algorithm for Finding Balanced Brackets Unbalanced