Alexandria University

Faculty of Engineering

Communication and Electronics Department

DS Sheet 3

Stacks

Name/ Mostafa Ahmed Mohamed Rashed
ID/ 19017528
Section/ 3

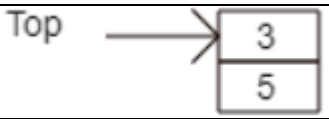
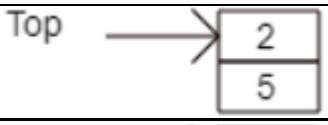
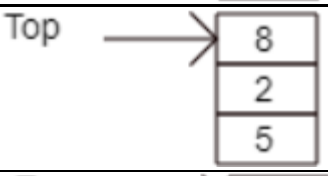
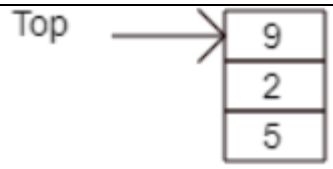
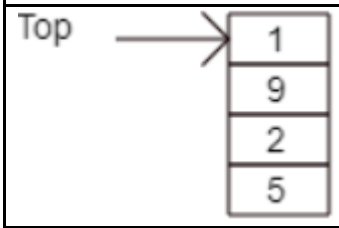
1. Suppose an initially empty stack S has performed a total of 25 push operations, 12 top operations and 10 pop operations. Three of the pop operations generated "Empty Stack Exception", which were caught and ignored.

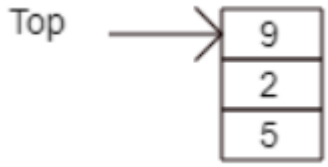
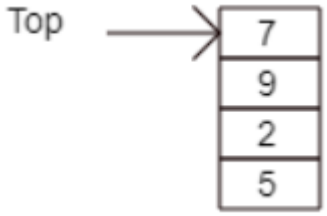
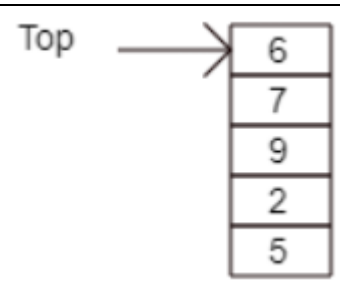
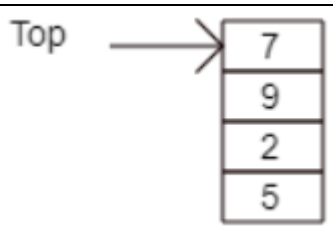
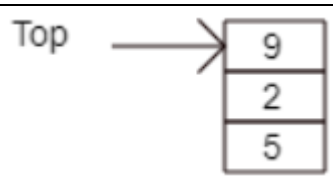
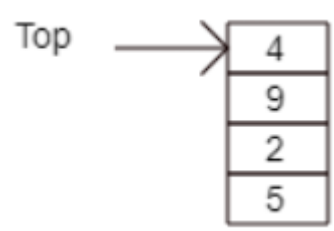
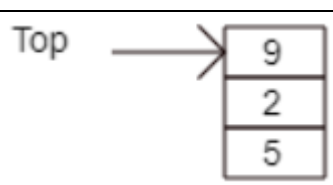
1. What is the size of S after performing the operations described above?
2. If this stack is implemented as an array, what is the value of the "top" data member of the Stack class?

1. Size → 25-(10-3) = 18
2. Top element → Size-1 = 17

2. Describe the output of the following series of stack operations:

```
push(5), push(3), pop(), push(2), push(8), pop(), push(9), push(1),
pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop().
```

Stack	Output
	
	
	3
	
	
	8
	
	

 Top 9 2 5	1
 Top 7 9 2 5	
 Top 6 7 9 2 5	
 Top 7 9 2 5	6
 Top 9 2 5	7
 Top 4 9 2 5	
 Top 9 2 5	4
 Top 2 5	9

3. Write an algorithm that returns the number of elements in a stack leaving it unchanged. (Assume that the stack Abstract Data Type provides only pop and push operations).

```
int stack_size(Stack stack)
{
Stack tempStack = stack;
int size = 0;
try{while(true){
  stack.pop();
  size++;}}
catch(Exception e) {return size;}}
```

4. Write an algorithm that uses a stack to determine if an HTML document is well-formed. (A well-formed HTML document should have all tags properly nested and all opened tags should have the corresponding closing tags).

```
Algorithm checkHTML(String html){
String[] tags = match(html , "<[^>]*>" ); //regex matches of html tags
Stack tempStack = new Stack;
For(String tag : tags){
If(!(tag.contains("/"))) tempStack.push(tag);
Else{ tag = tag.replace("/", "");
If(tag == tempStack.pop()) continue;
Else{ print("non valid html");  return false;}
}}
Return true;}
```

5. A palindrome is a word or a phrase that is the same when spelled from the front or the back. For example reviver and able was I ere I saw elba are both palindromes. Write an algorithm that uses a stack to determine if a word or a phrase is a palindrome.

```
isPalindrome(String word){
Stack tempStack = new Stack();
word = word.replace(" ","");
char[] wordChar = word.toCharArray();
int size = wordChar.length
for(int I = 0; I < size/2; I++) tempStack.push(wordChar[i])
for(int I = (size/2) +2; I < size; I++) {
if(!tempStack.isEmpt()){
 if(tempStack.pop() != wordChar[i] ) return false;
}}
Return true;
}
```

6. Write an algorithm that doing the following on the stack leaving it unchanged:

1. Return an identical copy of the Stack.
2. Return a reversed copy of the Stack.
3. Return a sorted copy of the Stack in descending order.
4. (Assume that the stack Abstract Data Type provides only `pop, push, peak, and isEmpty` operations).

1.

```
Algorithm reverseStack(Stack stack){
Stack tempStack = new Stack;
tempStack = stack;
return tempStack;
}
```

2.

```
Algorithm reverseStack(Stack stack){
Stack tempStack = new Stack;
While(!stack.isEmpty()){
tempStack.push(stack.pop());}
return tempStack;
}
```

3.

```
Algorithm sortedStack(Stack stack){
Stack tempStack = new Stack;
While(!stack.isEmpty()){
Int top = stack.pop();
While(!tempStack.isEmpty() && tempStack.peek() > top){
stack.push(tempStack.pop());}
tempStack.push(top);
}
Return tempStack;
}
```