CSCI2100C Tutorial2

ZHANG Xinyun

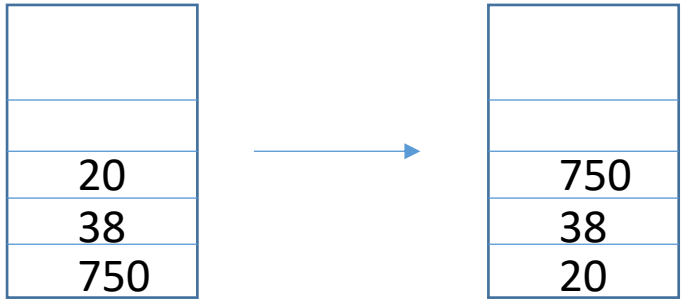
xyzhang21@cse.cuhk.edu.hk

- Exercise1 -> Reverse a stack
- Exercise2 -> Valid Parentheses
- Exercise3 -> Sort a stack

Exercise1-1

Problem Definition

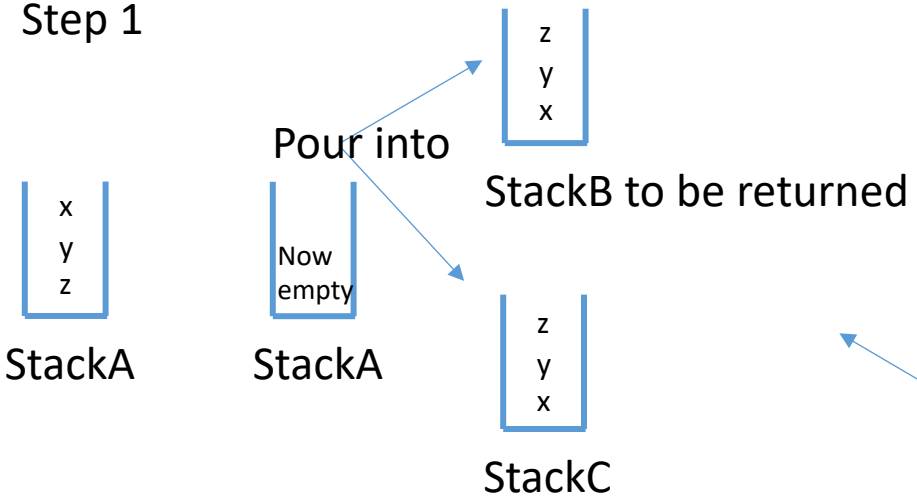
Write the C function ReverseStack1(). The function accepts a stackADT argument, and returns a stack that contains the same elements in the argument stack except that the order of the elements is reversed. The argument stack should be unchanged.



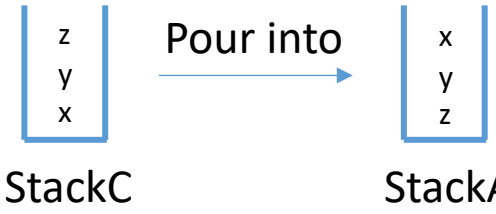
stackADT ReverseStack1(stackADT stack);

Answer

Step 1



Step 2 (why do we need this step?)

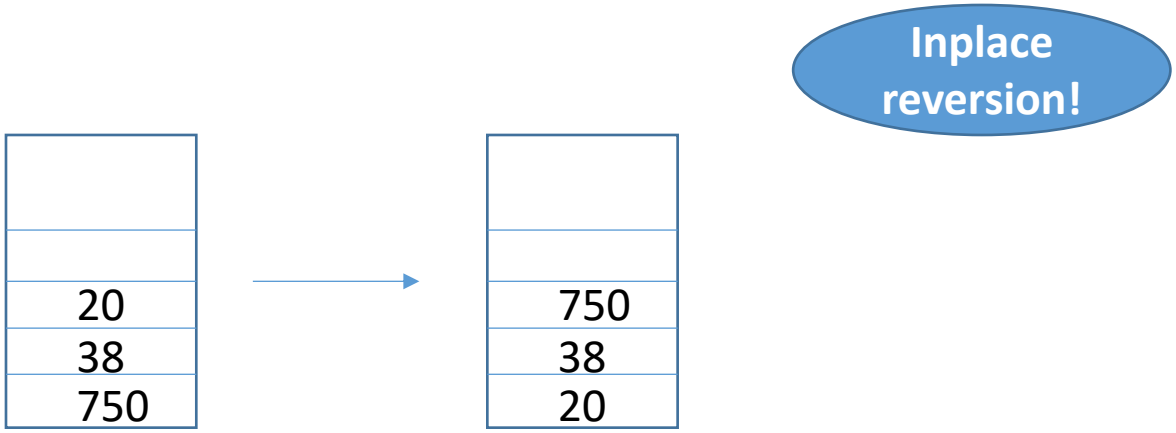



```
stackADT ReverseStack1(stackADT stack){
    stackADT stack_1, stack_2;
    stack_1 = EmptyStack();
    stack_2 = EmptyStack();
    int depth = StackDepth(stack);
    for(int i = 0; i < depth; i++){
        stackElementT element = Pop(stack);
        Push(stack_1, element);
        Push(stack_2, element);
    }
    for(int i = 0; i < depth; i++)
        Push(stack, Pop(stack_2));
    return stack_1;
}
```

Exercise1-2

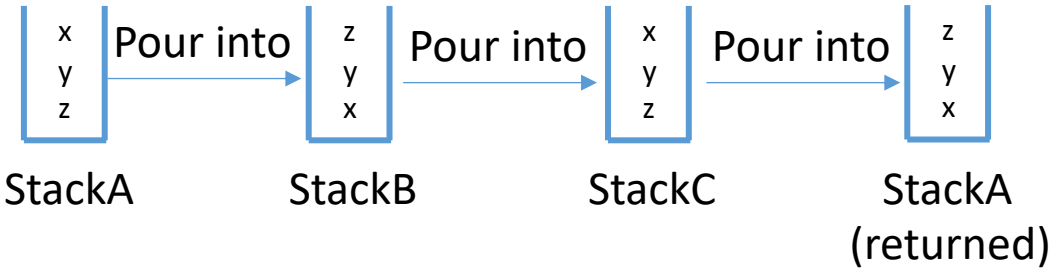
Problem Definition

Write the C function ReverseStack2(). The function accepts a stackADT argument, and reverses the elements stored in it.



void ReverseStack2(stackADT stack);

Answer Version1




```
void ReverseStack2(stackADT stack){
    stackADT stack_1, stack_2;
    stack_1 = EmptyStack();
    stack_2 = EmptyStack();
    int depth = StackDepth(stack);
    for(int i = 0; i < depth; i++)
        Push(stack_1, Pop(stack));
    for(int i = 0; i < depth; i++)
        Push(stack_2, Pop(stack_1));
    for(int i = 0; i < depth; i++)
        Push(stack, Pop(stack_2));
}
```

Optional: Answer V2 (recursion)

```
stackElementT RemoveTheLastElement(stackADT stack){
    stackElementT element = Pop(stack);
    if(StackIsEmpty(stack))
        return element;
    else{
        stackElementT lastElement = RemoveTheLastElement(stack);
        Push(stack, element);
        return lastElement;
    }
}

void RecursiveReverseStack(stackADT stack){
    if(StackIsEmpty(stack))
        return;
    stackElementT lastElement = RemoveTheLastElement(stack);
    RecursiveReverseStack(stack);
    Push(stack, lastElement);
}
```

Get the bottom element of stack and remove it from stack

Reverse the whole stack

Exercise2

Problem Definition

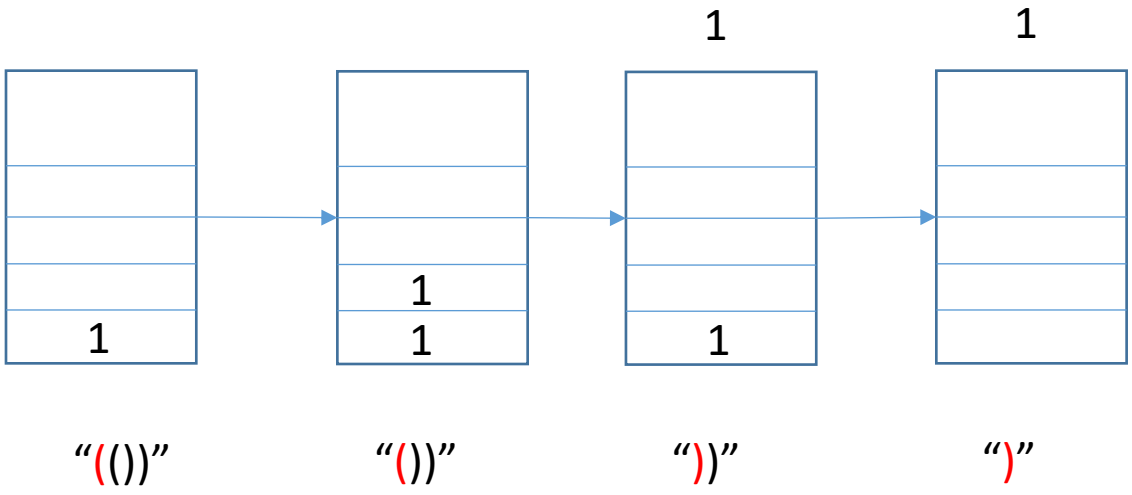
Given a string containing just characters ‘(’ and ‘)’, determine if the input string is valid, which means the brackets close in correct order. For example, “(()())” is valid but “)()()” is not. Write the c function int IsValid(char* s) to solve this problem. If string s is valid return 1 if not return 0.

Hint. You may consider to use a stack.

Answer

Key idea: for each right bracket ‘)’, there must be a left bracket ‘(’ to match it.

Solution:
for each ‘(’ we push one element to the stack; for each “)”, we pop one element out.



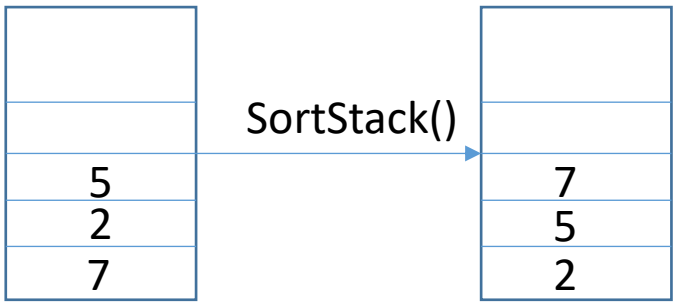

```
int IsValid(char* s){
    stackADT stack = EmptyStack();
    int len = strlen(s);
    for(int i = 0; i < len; i++){
        char c = s[i];
        if(c == '('
            Push(stack, 1);
        else{
            if(StackIsEmpty(stack))
                return 0;
            Pop(stack);
        }
    }
    return StackIsEmpty(stack);
}
```

Exercise3

Problem Definition

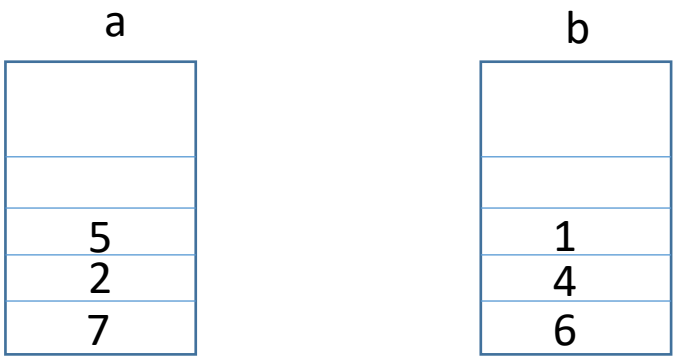
Given a stack s, sort the stack. The only additional data structure you can use is another stack.

Write the c function void SortStack(stackADT stack).



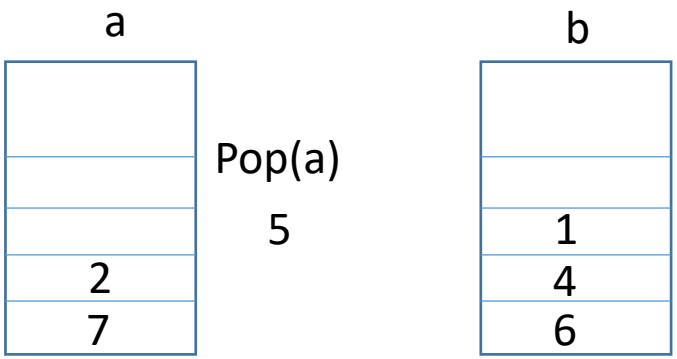
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the elements of a into b?



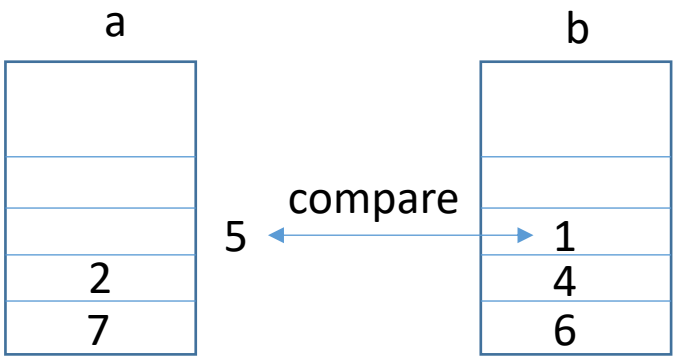
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



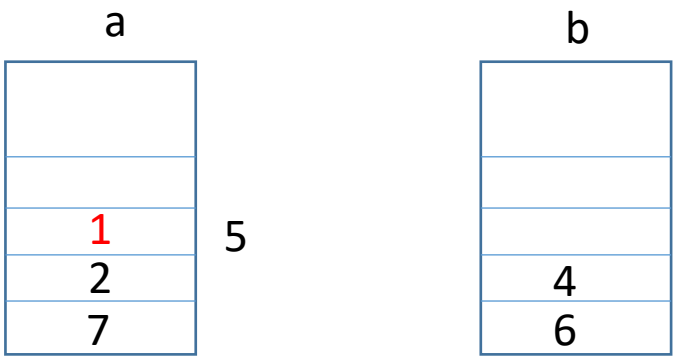
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



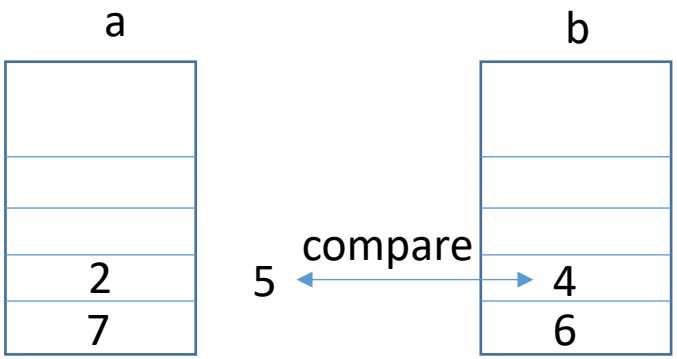
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



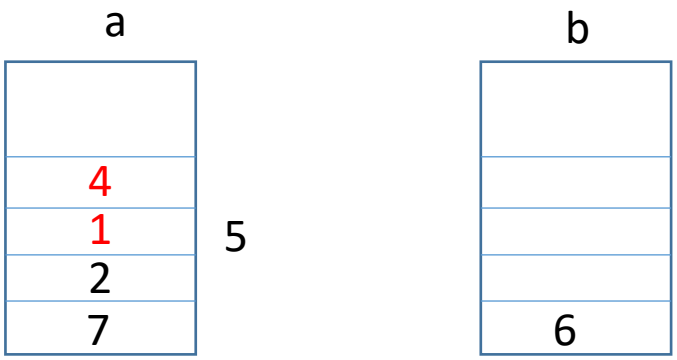
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



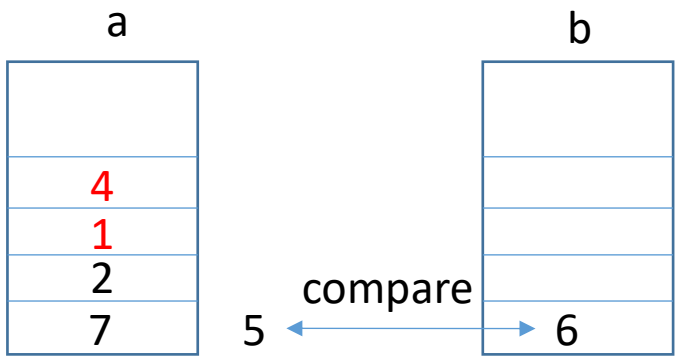
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



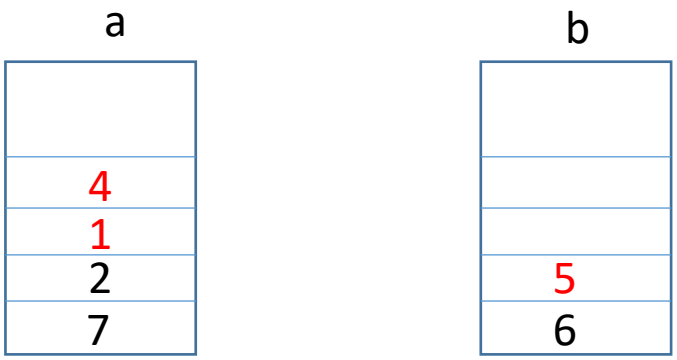
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



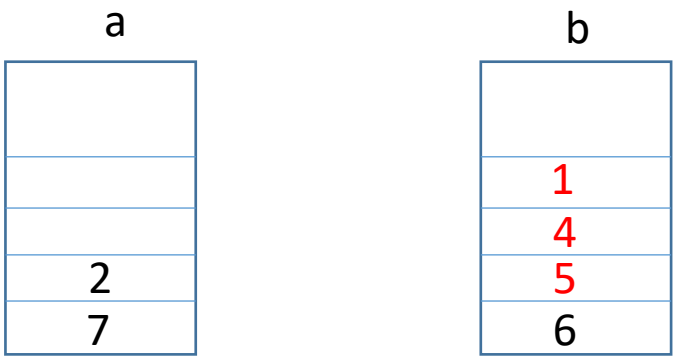
Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the Elements of a into b?



Answer

Consider two stacks a and b, suppose a is un-ordered and b is ordered, how to insert the elements of a into b?

```
stackElementT TopElement(stackADT stack){
    stackElementT element = Pop(stack);
    Push(stack, element);
    return element;
}

void SortStack(stackADT stack){
    stackADT stack_1 = EmptyStack();
    while(!StackIsEmpty(stack)){
        stackElementT topElement = Pop(stack);
        // insert the element to the ordered stack_1
        while((!StackIsEmpty(stack_1)) && (TopElement(stack_1) < topElement))
            Push(stack, Pop(stack_1));
        Push(stack_1, topElement);
    }
    int depth = StackDepth(stack_1);
    for(int i = 0; i < depth; i++)
        Push(stack, Pop(stack_1));
}
```