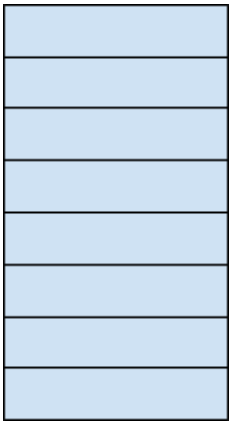
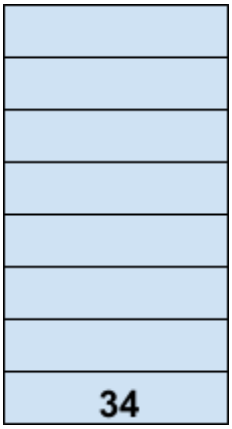
SCS 1201
Data Structures and Algorithms
Lab Sheet 02

1. Write down the output values of the following functions when performed on a stack data structure. (Stack size is 8) You need to use suitable diagrams to illustrate the state of the stack for different functions. At every stage write down the top value of the stack.
    a. isEmpty()



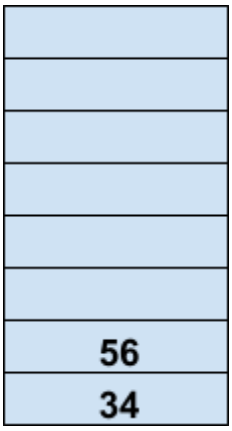
true, top=-1

    b. push(34)



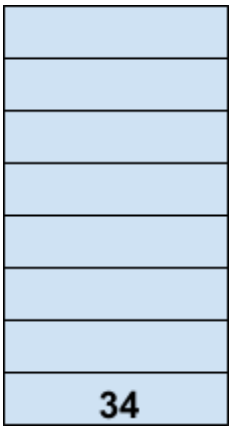
top=0

    c. push(56)



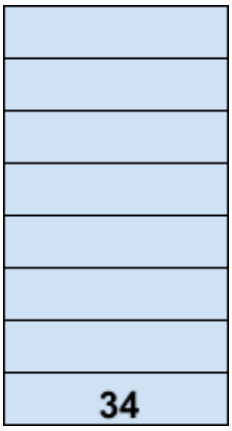
top=1

d. pop()



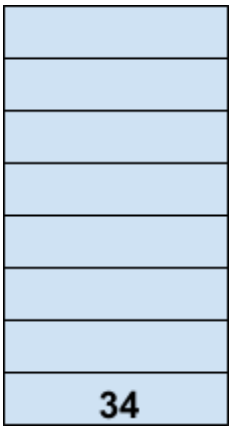
56, top=0

e. isEmpty()



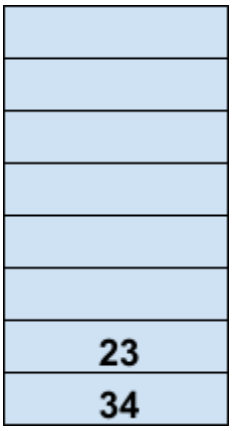
False, top=0

f. peek()



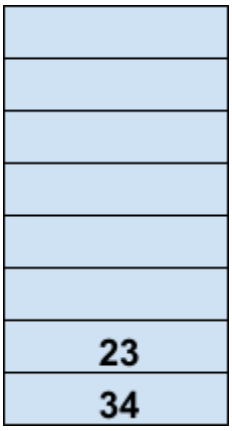
34, top=0

g. push(23)



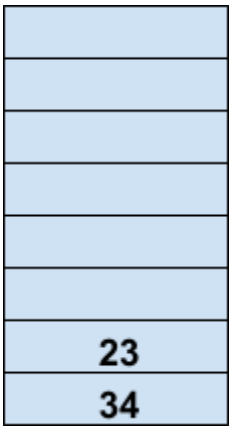
top=1

h. isFull()



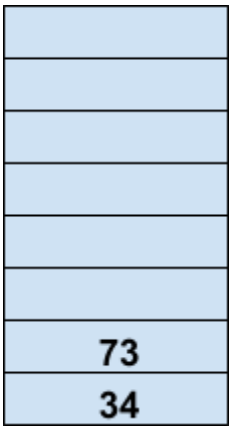
False, top=1

i. peek()



23, top=1

j. push(73)



top=2

2. Write down pseudo-codes to implement the **peek** and **isEmpty** operations of the stack data structure.

Answers

peek()

Step1: If top=**NULL**
Print "**Stack is empty**"
Jump to **step 3**
Step2: Return stack [top]
Step3: END

isEmpty()

```
Begin
        If top is less than 1
                Return true
        Else
                Retun false
        Endif
End
```

3. Write a pseudo-code to push the string “abcde” to a stack data structure.

Answers

```
String tempString=”abcde”
Stack tempStack

for(int i=0; i<5; i++)
{
        tempStack.push(tempString[i])
}
```

4. Write a pseudocode to print the above String’s characters in reverse order only using a stack data structure. You can use the **peek** and **isEmpty** functions.

Answers

```
While(( !tempStack.isEmpty() )
{
        Display (tempStack.pop())
}
```

5. A bracket is considered to be any one of the following characters: (,), {, }, [, or]. Two brackets are considered to be a match if an opening bracket occurs to the left of a closing bracket of the exact same type.So there are three types of matched pairs of brackets: [], {}, and (). Expression with brackets said to be balanced if all the brackets it includes are matched. Otherwise it is said to be unbalanced.

   E.g. for balanced brackets: []{}(), [({})]{}() and ({(){}[]})[]
   E.g. for unbalanced brackets: {[(])}, ({(){}[]))[] and {(){}

Given an expression string exp , write a pseudocode to examine whether the expression is balanced or unbalanced. (**Hint:** Consider you have almost implemented a stack with all the stack operations and initialized it with the name "stack".Write a pseudocode only to check whether the expression is balanced using basic stack operations. Consider expression exp only contains bracket characters and it does not contain any other alphanumeric characters.

E,g. for exp : {[()]}, {[(])}, [()]{}{[()()]()})

Answers

```
BEGIN Program
READ String exp
Initialize Stack stack
FOR each character in exp:
    IF character equals to "(" or "{" or "[":
        PUSH character to stack
    ElSE:
        SET current_character TO POP value of stack
        IF current_character EQUALS TO "(":
            IF character NOT EQUALS TO ")":
                OUTPUT "Expression is unbalanced"
                END Program

        IF current_character EQUALS TO "{":
            IF character NOT EQUALS TO "}":
                OUTPUT "Expression is unbalanced"
                END Program

        IF current_character EQUALS TO "[":
            IF character NOT EQUALS TO "]":
```

```
                    OUTPUT "Expression is unbalanced"
                    END Program
END FOR
OUTPUT "Expression is balanced"
END Program
```

6. Consider you have asked to develop a special stack with all the stack operations such as push(), pop(), isEmpty(), peek(), isFull() with one additional operation getMin() which return the minimum value of the special stack. State how you can implement this special stack.

Answers

- Declare a variable to store **minimum value (minValue).**
- Important thing to notice in the question is how we can handle the minimum value when the current minimum value is **removed (pop)**.
- To handle this we **push** (2 * pushed value – minValue) into stack instead of real value if this pushed value is lower than current minValue.
- Here (2 * pushed value – minValue) is always less than pushed **minimum value**. In the minValue we store real pushed values.
- So when **popping elements** we can identify these unusual values and perform the operation (minValue = 2 * minValue – value pop out) to get the next minimum value in the stack.

Operations:

Push(x) :
Inserts x at the top of stack.

- If the stack is **empty**, insert x into the stack and make **minValue** equal to **x.**
- If the stack is **not empty**, compare x with minValue. Two cases arise:
    - If x is **greater than** or **equal** to **minValue**, simply insert x.
    - If x is **less than** minValue, insert (**2 * x – minValue**) into the stack and make minValue equal to x.
        - For example, let previous minValue was 3. Now we want to insert 2. We update minValue as 2 and insert 2 * 2 – 3 = 1 into the stack.

Pop() :

Removes an element from top of stack.

- Remove element from top. Let the removed element be y. Two cases arise:
  - If y is **greater than** or **equal** to **minValue,** the minimum element in the stack is still minValue.
  - If y is **less than** minValue, the minimum element now becomes **(2*minValue – y)**, so update (minValue = 2*minValue – y).
  - This is where we retrieve the previous minimum from the current minimum and its value in the stack.
    - For example, let the element to be removed be 1 and minValue be 2. We remove 1 and update minValue as 2*2 – 1 = 3.

6.

a. Process of inserting an element in the stack is called _____.
   a) Create
   b) Push
   c) Evaluation
   d) Pop

Answers

b

Explanation: **Push** operation allows users to **insert** elements in the stack. If the stack is filled completely and trying to perform push operation stack – overflow can happen.

b. Process of removing an element from the stack is called _____.
   a) Create
   b) Push
   c) Evaluation
   d) Pop

Answers

d

Explanation: Elements in stack are removed using **pop** operation. Pop operation removes the top most element in the stack i.e. last entered element.

c. Pushing an element into a stack already having five elements and stack size of 5, then stack becomes.
   a) Overflow
   b) Crash
   c) Underflow
   d) User flow

Answers

a

Explanation: The stack is filled with 5 elements and pushing one more element causes a **stack overflow**. This results in overwriting memory, code and loss of unsaved work on the computer.

d. Consider the usual algorithm for determining whether a sequence of parentheses is balanced.The maximum number of parentheses that appear on the stack AT ANY ONE TIME when the algorithm analyzes: (()(()(()))) are:
   a) 1
   b) 2
   c) 3
   d) 4 or more

Answers

c

Explanation:

- In the entire parenthesis balancing method when the incoming token is a **left parenthesis** it is pushed into the stack.
- A **right parenthesis** makes a **pop operation** to delete the elements in the stack till we get left parenthesis as top most element.
- 3 elements are there in the stack before the **right parentheses** come. Therefore, the maximum number of elements in stack at run time is 3.

e. Which of the following real world scenarios would you associate with a stack data structure?
   a) piling up of chairs one above the other
   b) people standing in a line to be serviced at a counter
   c) offer services based on the priority of the customer
   d) tatkal Ticket Booking in IRCTC

Answers

a
Explanation:

- Stack follows **Last In First Out (LIFO)** policy. Piling up of chairs one above the other is based on **LIFO**, people standing in a line is a queue and if the service is based on priority, then it can be associated with a priority queue. Tatkal Ticket Booking Follows **First in First Out Policy.**
- People who click the book now first will enter the booking page first.

7.

a. The data structure required to check whether an expression contains balanced parenthesis is?
   a) Stack
   b) Queue
   c) Array
   d) Tree

Answers

a) Stack

b. Which data structure is used for implementing recursion?
   a) Queue
   b) Stack
   c) Array
   d) List

Answers

b) Stack