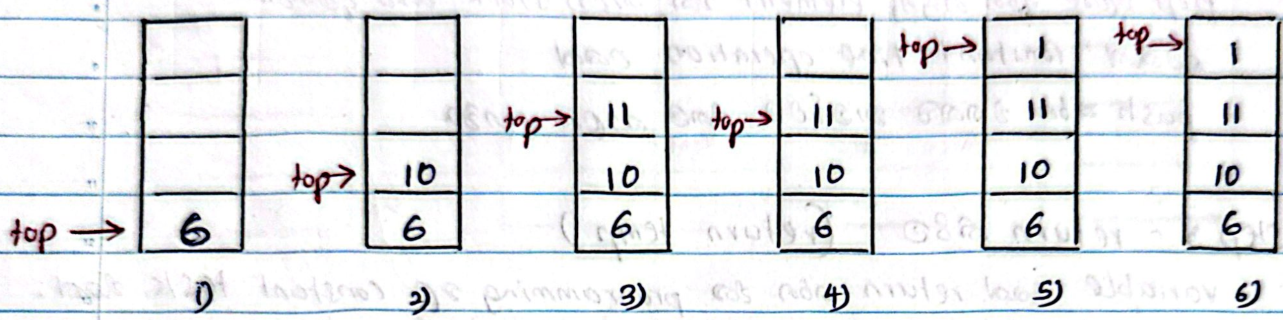
IT2070 - DSA - 2018 - October

a)
1) S1. push (6) ⟶ S1 stack එකට 6 push කරනවා S1 = [6]
2) S1. push (10) ⟶ S1 stack එකට 10 push කරනවා S1 = [6,10]
3) S1. push (11) ⟶ S1 stack එකට 11 push කරනවා S1 = [6,10,11]
4) S1. push (S1.pop()) ⟶ 11 remove කරලා return කරනවා. ඒ value එක නැවත push කරනවා
5) S1. push (1) ⟶ S1 stack එකට 1 push කරනවා.
6) S1. peek() ⟶ S1 stack එකේ top element එක බලනවා. Not Remove



b)(i)

```java
public class stackx {
  private char[] stack;
  private int top;

  public stackx (int size){
    stack = new char[size];
    top = -1;
  }

  public void push (char c){
    stack[++top] = c;
  }

  public char pop() {
    return stack[top--];
  }
```

```java
// implementing peek() using pop() and push()
  public char peek() {
    char temp = this.pop();
    this.push(temp);
    return temp;
  }
}
```

මෙහෙම implement කරද්දි pop() සහ push() method වල ආවත stack එකේ state එක එළියට නොපෙනෙන්න තැබීම වැදගත්.

peek() method එකේදී stack එකේ content එක එයාගේ එකෙන් වෙනස් වෙන්න එපා

(iv) step 1 - pop() method එක දාන්වම (char temp = this.pop();)

මෙහිදී stack එකේ top element එක remove කරනවා.

මෙය constant time (O(1)) යි.

එකක් remove කරන්න ඉවමන් එකම operation එකකින්.

step 2 - ~~pop~~ push() method එක දාන්වම (this.push(temp))

pop වලින් අරන් ගත්තු element එක නැවත stack එකට දානවා.

මේකත් constant time operation එකක්

push කරන එකට හැමවිටම එකම කාලය ගනවා

step 3 - return කිරීම (return temp)

variable එකක් return කරන එක programming වල constant task එකක්.

කිසිම loop එකක් නැහැ සහ recursion නොමැත.

→ සියලුම steps 03ම constant time operations

මුළු time complexity එක

$$O(1) + O(1) + O(1) = O(1)$$

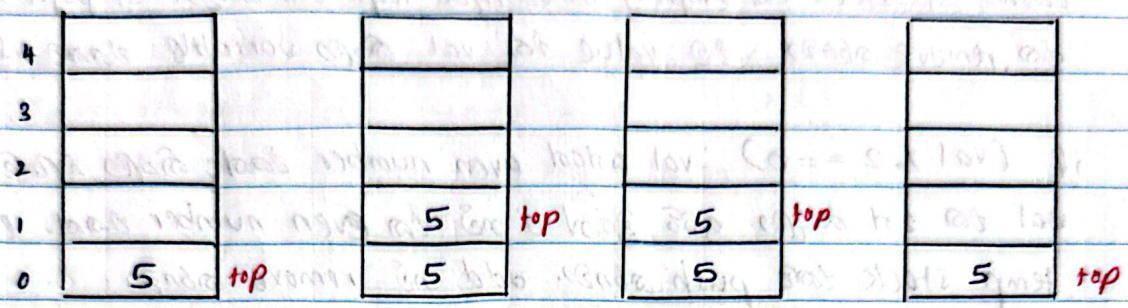
IT2090 - DSA - 2019 - October

a)(i) 1) S.push(s) ⟶ s stack එකට s push කරනවා

2) S.push(s.peek()); ⟶ top element එක කොපි කරලා අලුත push කරනවා

3) S.push(s.pop()); ⟶ s remove කරලා ඒ value එක ආපහු return කරනවා

4) S.pop(); ⟶ s remove කරලා දානවා.



(ii)
```
public void push (int j) {
    if (top == maxsize - 1)
        System.out.println ("Stack is full");
    else {
        top++;
        StackArray [top] = j;
    }
}
```

(iii) remove odd numbers and even numbers should remain in the stack.

```
public static void removeOddNumbers (Stack x s1) {
    Stack x temp = new Stack (s1.size());
    while (!s1.isEmpty()) {
        int val = s1.pop();
        if (val % 2 == 0) {
            temp.push(val);
        }
    while (!temp.isEmpty()) {
        s1.push (temp.pop());
    }
}
```

EDUCATION IS THE BEST INVESTMENT

removeoddnumbers කියන්නේ static method එකක්. මේ method එකට parameter එකක් තියෙනවා. s1 කියලා stack ක class එකේ stack object එකක්. ඊළඟට temp කියලා නව stack එකක් හදනවා. s1 stack එකේ size එකට සමාන ප්‍රමාණයක් තියෙන stack එකක්. මෙය අපිට even numbers තාවකාලිකව කරන්න දානට තියෙන stack එකට වෙනම temporary stack එකක්

ඊළඟට s1 stack එක empty නොවෙනතුරු loop එක යනවා. s1.pop() top element එක remove කරනවා. ඒ value එක val කියලා variable එකකට assign කරනවා,

if (val % 2 == 0) val කියන්නේ even number එකක්ද කියලා බලනවා.

val එක 2න් බෙදලා ඉතිරි වුනොත් 0 නම් ඒක even number එකක්. even නම් temp stack එකට push කරනවා. odd නම් remove කරනවා

No. .... Date. ....

IT2070 - DSA - 2022 - June

1) Similarity – Both stack and Queue are linear data structures that allow elements to be inserted and deleted. They follow a specific order for these operations and are often implemented using arrays or linked lists

Difference – stack follows Last in First out
Queue follows First in First out

2) use two stacks (stack objects are "s" and "s1")

Input : Stack [] = [1, 2, 3, 4, 5]
Output : Stack [] = [1, 2, 4, 5]
Input : Stack [] = [1, 2, 3, 4, 5, 6]
Output : Stack [] = [1, 2, 3, 5, 6]

```
public void deleteMiddle() {
    int count = 0;
    while (!s.isEmpty()) {
        s1.push(s.pop());
        count++;
    }

    int middle = count / 2;

    for (int i = 0; i < count; i++) {
        int val = s1.pop();
        if (i != middle) {
            s.push(val);
        }
    }
}
```

මෙහිදී මුල් stack එකේ (s) හැම element එකක්ම pop කරලා දෙවන stack එකට (s1) push කරනවා. ඒකේ අරගෙන තමයි stack එකේ element count එක ගණිත.

int middle = count /2 මේ line එකෙන් මැද index එක හොයනවා. count = 5 නම් middle = 2 (0 based index)

for (int i=0; i<count; i++) මෙහිදී s1 එකේ එක එක element pop කරලා i==middle නම් ඒක skip කරනවා. push කරන්නේ නෑ ඉතිරි ඒවා ආපහු s එකට push කරනවා.

EDUCATION IS THE BEST INVESTMENT

Scanned with CamScanner

No. Date.

IT2070 - DSA - 2022 - November

a)

int[] stackArray
int maxSize
int top
void push(double j)
double pop()
double peek()
boolean isEmpty()
boolean isFull()

(i) myStack.push(6);
(ii) myStack.peek();
(iii) myStack.push(2);
(iv) myStack.pop();
(v) myStack.pop();

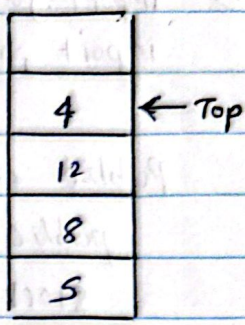
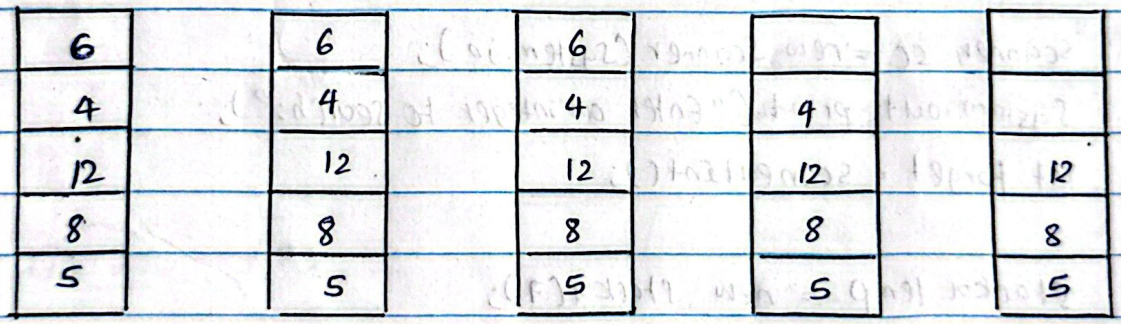


Figure 1



(i) (ii) (iii) (iv) (v)

b)
```
import java.util.Scanner;
import java.util.Stack;

Public class StackSearchExample {
    public static void main (String[] args) {
     Stack st = new Stack(7);
      st.push(20);
      st.push(93);
      st.push(12);
      st.push(37);

      Scanner sc = new Scanner (System.in);
      System.out.print("Enter an integer to search:");
      int target = sc.nextInt();

      Stack temp = new Stack(7);
      boolean found = false;

      while (!st.isEmpty()){
       int val = st.pop();
       if (val == target) {
          found = true;
       }
       temp.push (val);
      }
      while (!temp.isEmpty()) {
        st.push (temp.pop());
      }
      if (found) {
        System.out.println (target + " is found in the stack.");
    }else{
        System.out.println (target + " is Not found in the Stack.");
    }
}
```

No. .... Date. ....

IT2070 - DSA - 2023 - June

a)

	Insert	Delete	Retrieve
stack	push() - at the top	pop() - from the top	peek() - get top element
linear queue	enqueue() - at the rear	dequeue - from the front	check front without removing
Circular queue	enqueue() - at the rear (wraps around)	deque - from the front (wraps around)	access front / rear using index

b) (i) Most popular data structure is Array

The process of retrieving the element at the top of stack without removing it. it is called peek

A stack can be implemented using array or linked list

In a stack, if the top pointer is equal -1, then the stack is empty

The time complexity of the push and pop operations on a stack implementation are:-

push O(1) and pop O(1)

→ Array යනු සරල හා වේගවත් data structure එකක්. සරලව භාවිත කළ හැකිය. ඕනෑම programming භාෂාවක තිබේ. index එකක් භාවිතයෙන් ඕනෑම එකක් වේගයෙන් (O(1)) ලබාගත හැක.

→ peek() ක්‍රියාවෙන් stack එකේ ඉහළම අගය ලබාගන්න පුළුවන්.

→ Array - නියත ප්‍රමාණයක් සහිතයි. index එකෙන් අගය ලබාගත හැකිය.

linked list ගතික ප්‍රමාණය ගැළපෙන විදියට වෙනස් කළ හැක.

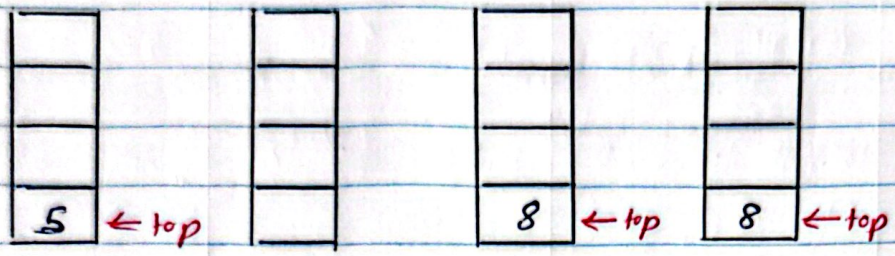
→ Array භාවිත කරමින්, stack එකේ top = -1 යන අගයෙන් stack එක තුළ කිසිදු අගයක් නොමැති බවක් පෙන්වයි. stack එකේ ආරම්භක අගයයි.

→ push() එකේ ඇතුළු වන විට top++ කරලා ඊට පසු value එක දමන නිසා O(1)

pop() එකේ ඉවත් කරන විට top-- කරලා අගය return කරන නිසා O(1)

IT2070 - DSA - 2023 - November

a) Stack - undo feature in a text editor (MS word / Notepad)
Queue - Print queue in a Printer



(i) (ii) (iii)

(i) stack.pop()
(ii) stack.push(8)
(iii) stack.peek()

(i) The "top" is -1 when stack is Empty.
(ii) The no of items in a stack can be obtained from (top+1)
(iii) A stack can be implemented using array or linked list
(iv) "Peek front" operation of a linear queue and circular queue are same
(v) The time complexity of the insert operation of a queue implementation is O(1)

Page No 1 Index No Date

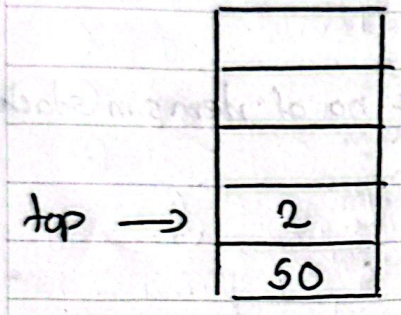
Tutorial 1 - Stacks

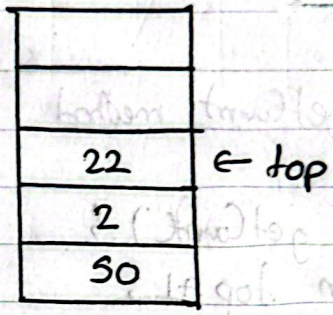
Q1

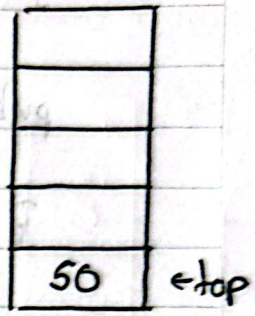
(a) Consider the following stack and draw the stack frames after executing each statement given below.

```
int a = 22, b = 44
```

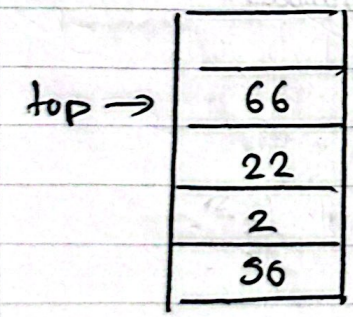
(i) theStack.push (2); (ii) theStack.push (a);

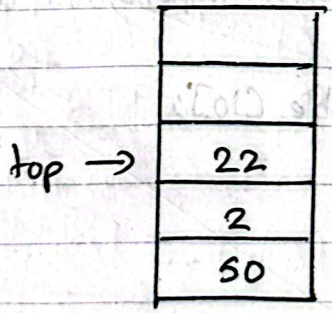


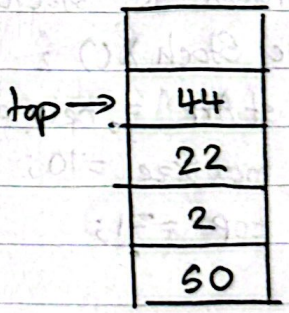


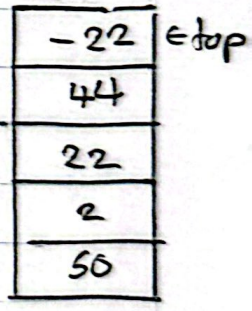


(iii) theStack.push(a+b); (iv) theStack.pop(); (v) theStack.push(b); (vi) theStch.push(a-b);









Q2 Consider the stackX class given below

```
int top
int maxSize
int stackArr[]
```

```
StackX (int size)
void push (int no)
int pop()
boolean isEmpty()
boolean isFull()
int getCount()
```

Ruhunu

(i) Implement isEmpty() and isFull() methods of this stack class.

```
public boolean isEmpty() {
    return (top == -1);
}

public boolean isFull() {
    return (top == maxSize - 1);
}
```

(ii) Implement getCount method to return the no of items in stack.

```
public int getCount() {
    return top + 1;
}
```

Q3 constructor of stack class implemented as follows

```
public StackX() {
    stArr = new double [10];
    maxSize = 10;
    top = -1;
}
```

(i) Mention one disadvantage of having the above constructor.

Size of stack is hardcoded and not flexible.

(ii) Rewrite the constructor to avoid the disadvantage mentioned above.

```
public StackX(int size) {
    maxSize = size;
    stArr = new double [maxSize];
    top = -1;
}
```

Q4

i A stack class has been implemented with push(), pop(), peek() methods It is

a. used to store characters. Write a code segment to insert following characters to a 'myStack' object created from the Stack class 'g', 't', 'o', 'p'

```
public static void main (String[] args) {
        StackX myStack = new StackX (5);
        myStack. push ('g') ;
        myStack. push ('t') ;
        myStack.push('o');
        myStack.push('p');
}
```

ii Write code segment to display all the values in a stack by removing them.

~~public void~~

```
while (! myStack.isEmpty()) {
        System.out.println(myStack.pop());
}
```

iii What is the result of section ii above

p
o
t
g

Q5: Fill in the banks.

(i) Most popular data structure is Array

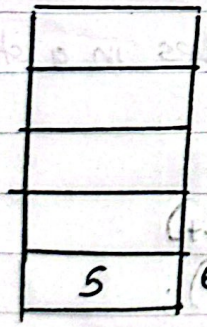
(ii) The process of retreiving the element at the top without removing it is called peek

(iii) A stack can be implemented using Array or Linked list

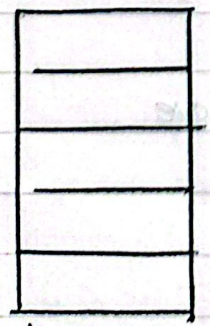
(iv) In a stack, if the "top" pointer is equal -1 then the stack is empty

(v) The time complexity of the push & pop operations on a stack implementation are; push - O(1) and pop - O(1).

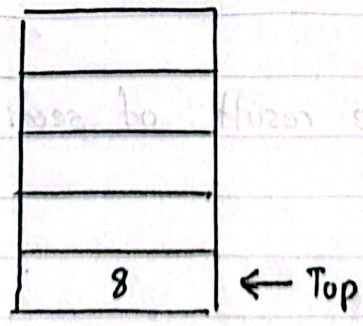
Q6: Consider the initial stack frame of a stack given below.



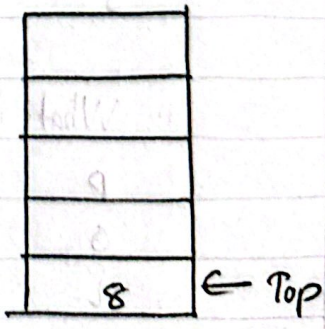
Below stack frames are obtained after executing 3 operations one after another to the above stack frame. Write down the operations.



stack.pop();



stak.push(8);



stack.peek();

Lab Exercise 1 - Stacks

Question

(a) Implement Stack class to store characters. Identify data members required & implemented the below methods with the constructor.
Push(), Pop(), Peek(), isEmpty(), isFull()

```
public class Stack {
    private int top;
    private int[] intStack;
    private int maxSize;

    public Stack (int size) {
        this.maxSize = size;
        intStack = new int[maxSize];
        top = -1;
    }

    public boolean isEmpty() {
        return (top == -1);
    }

    public boolean isFull() {
        return (top == maxSize-1)
    }
}
```

b) Develop an application to

i) Insert characters to the stack using push() method.

ii) Remove content of the stack and display them.

iii) Comment on the order of insertion to the stack and the order of removal from the stack.

i)
```
stack.push('a');
stack.push('b');
stack.push('c');
```

ii)
```
while(!stack.isEmpty()) {
    System.out.print(stack.pop() + " ");
}
```

iii) stack follows LIFO (Last In, First Out) principle.

c) Use the stack class created above and check whether a user entered expression is correctly paranthesized.

```
public static boolean isValidParantheses(String expr) {
    Stack stack = new Stack(expr.length());
    for (int i=0; i < expr.length(); i++) {
        ch = expr.charAt(i);
        if (ch == '(')
            stack.push(ch);
        else if (ch == ')') {
            if (stack.isEmpty())
                return false;
            stack.pop();
        }
    }
    return stack.isEmpty();
}
```