Assignment - 2

Name :- J. Surendra

Regd No :- 192324050

Subject :- Data structure

Sub code :- CSA0389

Date of Sub :- 29/07/24

Write Pseudocode for stack operations.

1. Initializestack():
   Initalize necessary variable Dstructures represent the stack.

2. Push():
   if stack isfull:
      Printf("stack overflow")
   else:
      add element to the top of the stack
      increment top Pointer.

3. Pop():
   if stack is empty:
      Printf("stack underflow")
      return null(or appropriate error value)
   else:
      remove and return element from the top of the stack
      decrement end Pointer.

4. Peek():
   if stack is empty:
      Printf("stack is empty").
      return null(or appropriate error value)
   else:
      return element at the top of the stack.

5. isEmpty():
   return true: if top is -1 (stack is Empty)
   Otherwise, return false.

6. isfull():
   return true, if top is Equal to maxsize -1 (stack is full)
   Otherwise, return false

Explanation of the Pseudo code

* Initializes the necessary variables (or) data structures to represent a stack.
* Adds an element to the top of the stack. checks if the stack is full before pushing.
* Removes and returns the element from the top of the stack. Checks if the stack is empty before popping.
* Returns the element at the top of the stack without removing it.
* checks if the stack is full by comparing the pointer or equivalent variable to the maximum size of the stack.

2

Describe the concept of Abstract data type (ADT) and how they differ from concrete data structures. Design an ADT for a stack and implement it using arrays and Linked List in c. Include operations like Push, Pop, Peek, is empty, is full and Peek.

Abstract Data Type (ADT)

An Abstract Data type is a theoretical model that defines a set of operations and the semantics of those operations on a data structure, without specifying how the data structure should be implemented. It Provides a high level description of what operations can be performed on the Data and what constraints apply to those operations.

Characteristics of ADT :-

- operations

Defines a set of operations that can be performed on the data structure.

- Semantics

Specifies the behaviour of each operation.

- Encapsulation.

Hides the implementation details, focusing on the interface Provided to the user.

ADT for stack :-

A stack is a fundamental data structure that follows the Last In first out Principle. It supports the following operations

- Push → Adds an element to the top of the stack.
- Pop → Removes and Returns the element from the top of the stack.

- Peek → Returns the element from the top of the stack without removing it.
- Is Empty → checks if the stack is empty.
- Is Full → checks if the stack is full.

Concrete Data structures :-

The Implementations using arrays and Linked Lists are specific ways of implementing the stack ADT inc.

How ADT differ from concrete Data structures :-

ADT focuses on the operations and their behaviours, while concrete data structures focus on how those operations are realized using specific Programming Constructs.

Advantages of ADT :-

By separating the ADT from its implementation, you achieve modularity, encapsulation and flexibility in designing and using data structures in Programs. The Separation allows for easier maintenance, code reuse, and abstraction of the complex operations.

Implementation

```
#include<stdio.h>
#define MAX_SIZE 100
typedef struct {
        int items[MAX_SIZE];
        int top;
} stackArray;
int main(){
    StackArray stack;
    stack.top=-1;
    stack.items[++stack.top]=10;
    stack.items[++stack.top]=20;
    stack.items[++stack.top]=30;
if(stack.pop!=-1){
    Printf("Top element: %d\n", stack.items[stack.top]);
}else{
    Printf("stack is empty!\n");
}
if(stack.top!=-1){
   Printf("popped element: %d\n", stack.items[stack.top--]);
} else{
   Printf("stack underflow!\n");
}
if(stack.top!=-1){
    Printf("popped element: %d\n", stack.items[stack.pop--]);
} else{
```

```
    Printf (" Stack underflow : \n");
}
if (stack.top != -1){
    Printf ("Top element after pops : %d \n", stack.items
                                                [stack.top]);
} else {
    Printf ("stack is empty : \n");
}
return 0;
}
```