Remarks _____ Returned on _____

Write an algorithm on a stack

1] Algorithm for add element at the end of stack.

Procedure PUSH (A, S, top, ele)

This procedure add element 'ele' at the end of the stack. 'A' A(1:S) is an array. 's' is maximum size of array. 'top' is the last element in stack.

Declaration :-

```
Global integer A(1:S), S, top
integer ele
    if (top == S), then
        print ("stack is Full")
        return
    endif
    top ← top+1
    A(top) ← ele
END_PUSH
```

2] Algorithm for pop/remove element from end of stack.

Procedure POP (A. top)

Description - This procedure pop element from the stack. A(1:S). 'A' is an linear array. 'top' is the last element present in a stack 'A'.

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

Declaration:-

```
Global integer A(1:s), top
    if (top = 0), then
        print("stack is empty")
        return (NULL)
    endif
    ele ← A[top]
    top ← top - 1
    return (ele)
END_POP
```

3] Algorithm for display all elements present in stack.

Procedure Display (A, top)

Description:- This procedure display all the elements currently present in stack A(1:s). 'A' is the linear array with maximum size 's', 'top' is the last element currently present in stack A(1:s)

Declaration:-

```
Global integer A(1:s), top
    Local integer i
        if (top = 0), then
            print("stack is empty")
            return;
        else
            for i ← 1 to top by +1 do
                print (A(i))
            repeat
        endif
END_Display.
```

Remarks

17] Write algorithm to perform PUSH operation in linked list.

Procedure, PUSH (top, data, next, ele)

① Description:-

This procedure add neco node in stack

using linked list - 'top' is the pointer pointing to the first node in the list which is initially set NULL. 'data' is a variable to hold the information to each node. 'next' is pointer pointing to the next node in the list its store the address of next node to point it. The 'next' pointer hold the null value in last node of the list. 'AVAIL' is a list of free nodes which is set to null. if no free node is available. 'ele' is element which is add to be.

② Declaration:-

Global pointer top, pointer next
integer data
praramter integer ele.

Algorithm:-

if AVAIL = NULL, then
print("stack is full")
else

Incomplete for :
1) Algorithm
2) Flow Chart
3) Programme Listing
4) Results
5) Comments

Remarks Returned on

1] Write a program to list all the elements in stack.

Procedure List_ALL (top, data, next)

Description :-

This procedure display all nodes from stack organized using Linked list. where 'top' is a pointer pointing to first node. in the list. which is initially set to null. 'Data' is variable part of node which ahold the information 'next' is a pointer pointing to next node in list by storing the address of next node which is eset to NULL of the last node in the list. 'Avail' list of free nodes.

Declaration :- Global integer data, pointer top, pointer next.

Algorithm :-

```
if top = NULL, then
    print("stack is empty")
else
    ptr ← top
    while ptr ≠ NULL, do
        print (ptr → data)
        ptr ← ptr → next
    repeat
endif
```

END List_ALL

```
        NEW <- DELETE (AVAIL)
        NEW -> data <- ele
        NEW -> next <- NULL
        NEW -> next <- top
        top <- NEW
    endif
END_PUSH
```

2] Write algorithm to perform POP operation in linked list.

Procedure POP (top, data, next)

Description:- This procedure delete node from stack organized using linked list from the first position. 'top' is a pointer pointing to the first node in the list which is initially set to NULL. 'Data' is a variable part of node which hold the information. 'next' is a pointer pointing to the next node in the list by storing the address of next node ~~in the~~ which is set to null of the last node in list. 'Avail' is a list of free nodes.

Declaration:- Global pointer top, pointer next
parameter integer ele.
Local pointer temp.

Algorithm:-

```
if top = NULL, then
    print ("stack is empty")
    return (NULL)
else
    ele <- top -> data
    Temp <- top
    start <- top -> next
    ADD (AVAIL) <- Temp
    return (ele)
endif
END_POP
```