Indian Institute of Technology (ISM), Dhanbad
Department of Computer Science and Engineering

Data Structures Lab(CSC204)

Assignment 4

Full Marks: 50

1 The Problem

Consider a scenario where we want to build a stack which is implemented using array but at the same time when the "stack overflow" condition arises, it can add another array of the same size. This is going to be a combination of array and list implementations of a stack. To implement this stack you have to do the following:

1. You have to define a structure to define a node of a singly linked list with only difference that instead of a single integer, each node will hold an array of integers. The size of the array n should be user defined. (**5**)

2. Your stack should have two variables to keep track of top, 1) *link_top*, 2) *stack_ top*. The *stack_ top* variable will keep track of the location of top element and the *link_top* variable will point to the list node on which the top element resides.

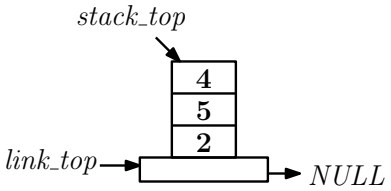
3. *expand_push(S1, x):* This operation will push an element at the top of the stack like normal stack. If the stack is full, a new list node will be added to the front of the list with same array size given by the user. Then you have to insert x in the new node. You have to update your top variables accordingly. (**10**)

4. *shrink_pop(S1):* This operation will extract the top element from the stack and return that value. When the last element of a node is extracted, the function will free that node of the list and update all the top variables accordingly. The "stack underflow" condition occurs when the *link_top* points to NULL. (**10**)

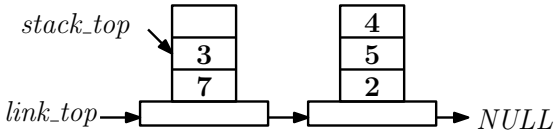
5. *elements(S1):* This function will return the total number of elements available in the stack. You are not allowed to store this number separately. (**10**)

6. *print_stack(S1):* This function will print all the elements of the stack. (**10**)

Overall design and organization of the program will carry 5 marks.



(i)



(ii)

2 Example Scenarios

(i) The stack *S1* after *expand_push(S1, 2)*, *expand_push(S1, 5)*, *expand_push(S1, 4)* is shown in figure (i). Here, $n = 3$.
(ii) *S1* after *expand_push(S1, 7)*, *expand_push(S1, 3)* is shown in figure (ii).
(iii) The result of *elements(S1)* at this point is 5.
(iv) Output of the *print_stack* at this point is as follows:
Array1: 3, 7
Array2: 4, 5, 2
(v) *S1* after two *shrink_pop(S1)* will be the figure shown in (i).