Date   Page

Day - 141

Stack - 7

* Implement two stack in an array :

⇒ So, we can do that, we will divide the array in two parts –

⇒ One for first stack & second for another stack.

⇒ But if we want to insert an element and that half of array is full but another half is empty, we can't insert the element into it.

⇒ So, this approach will not work.

⇒ Now, we will take two pointers top 1 & top2.

⇒ top 1 at starting & top 2 at end.

⇒ We will use top1 for stack 1 & top 2 for stack2.

⇒ If any element wants to come in ~~top~~ stack 1 than first we will increase the top1 & then push the element.

⇒ Same for stack 2 & in place of increasing, we will decrease by 1.

Code

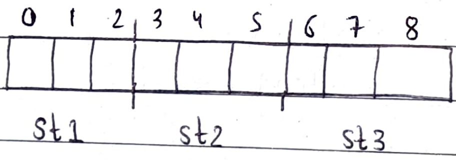
```
class  Nstack{
        public:
        int *arr;
        int top1, top2;
        int size ;
        Nstack (int n){
            arr = new int[n]; size = n;
            top1 = -1, top2 = n;
        }

        void  push1 (int x){
            if( top1+1 == top2)
                return;
            top1++;
            arr[top1] = x;
        }

        void push2 (int x){
            if( top2-1== top1)
                return;
            top2--;
            arr[top2] = x;
        }

        int pop1(){
            if(top1==-1)
                return -1;
            int ele = arr[top1];
            top1--;
        }   return ele;
```

```
    int pop2(){
       if( top2 == size )
            return -1;
       int ele = arr[top2];
       top2++;
       return ele;
    }
}
```

* N stack in an array:

⇒ There will be 'N' no. of stacks.

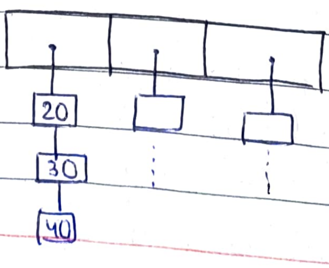
⇒ Suppose N = 3,



⇒ This approach is not the best approach.

⇒ In the starting, suppose you don't have any array & you only have 9 space & three stacks to implement.

⇒ Now, take an array of N size ~~that~~ and it will store address of stack top.



Date Page

=> we will take size variable to check available space.

=> If space is available then we will push element otherwise not.

=> Now, according to our question, we have to store element in array.

Code

```
class  Node{
        public:
        int index;
        Node * next;

        Node(int x){
            index = x;
            next = NULL;
        }
}

class  Nstack{
        public:
        int *arr;
        Node ** Top;
        stack<int> st;

        NStack(int N, int S){
            arr= new int[N];
        Top =  new Node*[N];
        for(int i=0; i<N; i++){
              Top[i] =  NULL;
        }
        for(int i=0; i<S; i++)
              st.push(i);
    }
        bool push(int x, int m){
             if( st.empty())
                   return 0;
             arr[st.top()] = x;
             Node *temp = new Node(st.top());
             temp->next  = Top[m-1];
             Top[m-1] =  temp;
             st.pop();
             return 1;
      }
      int pop( int m){
            if( Top[m-1]== NULL)
                  return -1;
   st.push <-                      arr
(Top[m-1]->index);  int element = [Top[m-1]->index];
            Top [m-1]  = Top[m-1]->next;
            return element;
      }
```