Stack Algorithms.

1) Create Stack. – allocate memory for "head"
Initialize the "head" (metadata)

2) PUSH Stack – insert an element on top.
(allocating memory)
– modify "head".
– other ptrs.

insert in
- empty stack.
- stack with data
- full stack. (results in overflow)

fails

3) Pop Stack :- deletes and returns the top element
(empty stack → underflow)

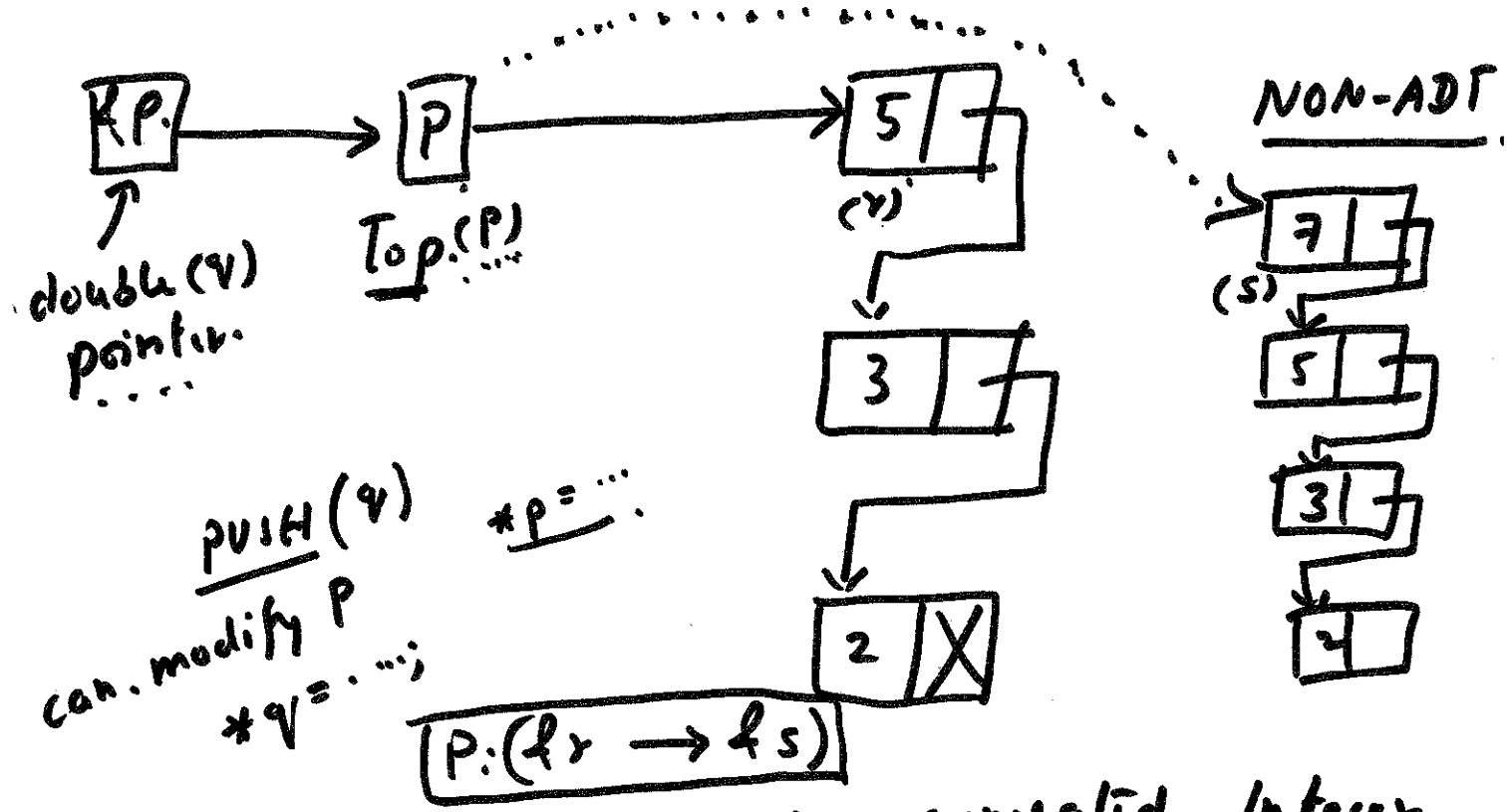
4) Stack Top : returns the top element (without deleting)

5) Empty Stack : determine if stack is empty.
(Data hiding : application/user may not have access to "head")

6) Full stack : determines if no more memory available.

7) Stack Count : returns # elements in stack.

8) Destroy Stack : delete all elements and "head" (free memory)



PUSH(q) *p = ...

can modify P

*q = ...;

rand() - returns randomly generated Integer b/w 0 and ~~MAX~~ MAXINT.

↓

% 26

↓

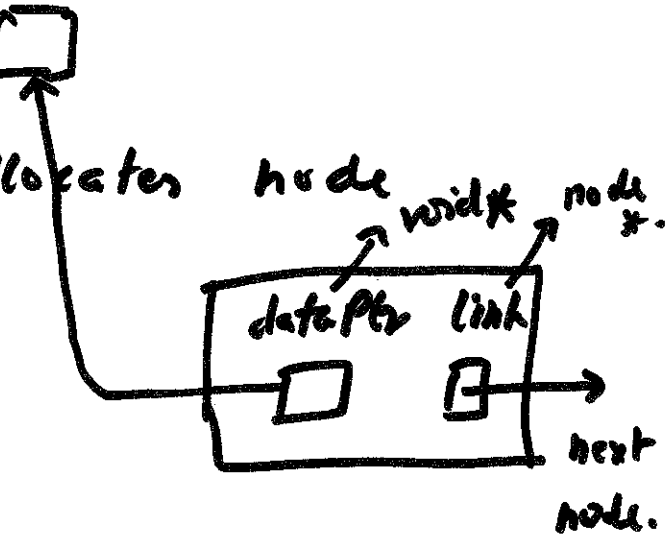
0 .... 25 + 'A'

```
char c = 'A' + 3;      /* c = 'D' */
printf("%d", c);       /* 68 */
printf("%c", c);       /* D */
```

ADT implementation

user allocates. data

stack implementation allocates node



Test Review.

T/F. Multiple choice. short Qn.

Pseudocode, Design., Code given → find errors.

① stack is a LIFO DS? T/F.

② — AB is in postfix notation?

③ Recursion is always preferred over iteration.

What is ADT. what feature in C are helpful.

efficiency — Big O notation.

← for (i=0; i<n; i++)

Q. Write C code to allocate memory for 10 structures of type A.

```
struct A {
    char b;
    int c;
    float d;
};
```

Q. Write pseudocode to print contents of the stack from top to bottom.

Q. Find error in given C code.

```
int i;
char str[100];
for (i = 0; i < 10; i++)
        str[i] = 'A' + i;
printf ("%s \n", str);
```

→ str[i] = '\0';

Q. Write a recursive f^n (pseudocode) to compute a^n (a is an integer)

f(a, n).

Base case : n = 1 → return a

General case : n > 1 → return a * f(a, n-1)

Q. Find <u>error</u>. in pseudocode.

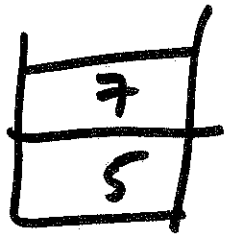
```
line nt   int fun(int x)
          {  if (x < 8)
fun(6)          return(x + fun(x));
                                  fun(x+1)
           else
                return 1;
          }
```

HW problems.

fun(4, 5) = ?.

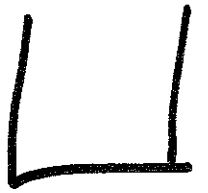
Given a fun(x, y)
{ . . . .
}

Q.



S1

pop stack(S1, x)
pop stack(S1, x)
push stack(S2, x)



S2

S1

S2