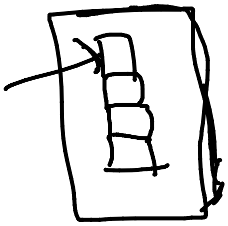
CLASS - 24

Stacks, Queue, Deque (Stack + queue + vector)

begin(), end(), random access

STL

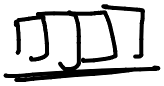


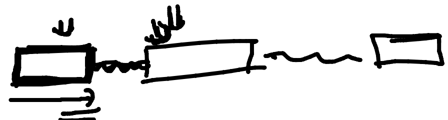
- Top
- Push
- Pop
- empty()

- front
- push
- pop
- empty()

- Push back
- Push - front
- pop - back
- pop - front
- insert
- empty()

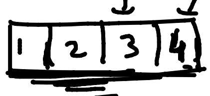
insertion of random access



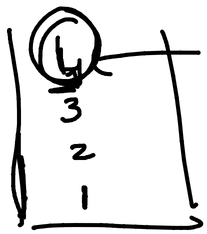


Array implementation

last, last



top => a[last]
last--;

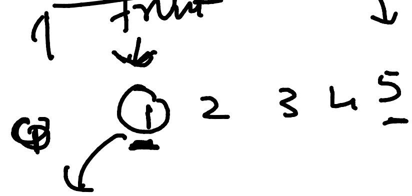


last

Queue

front

last / rear



front++;
return a[front];

a[last+1] = n

last++;

Single array

front

last

last+1

last--;

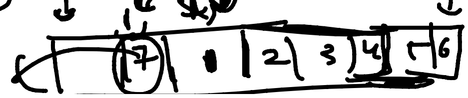
front--;

- Push - back()
- Pop - back()
- Push - front()
- Pop - front()

500, front, rear

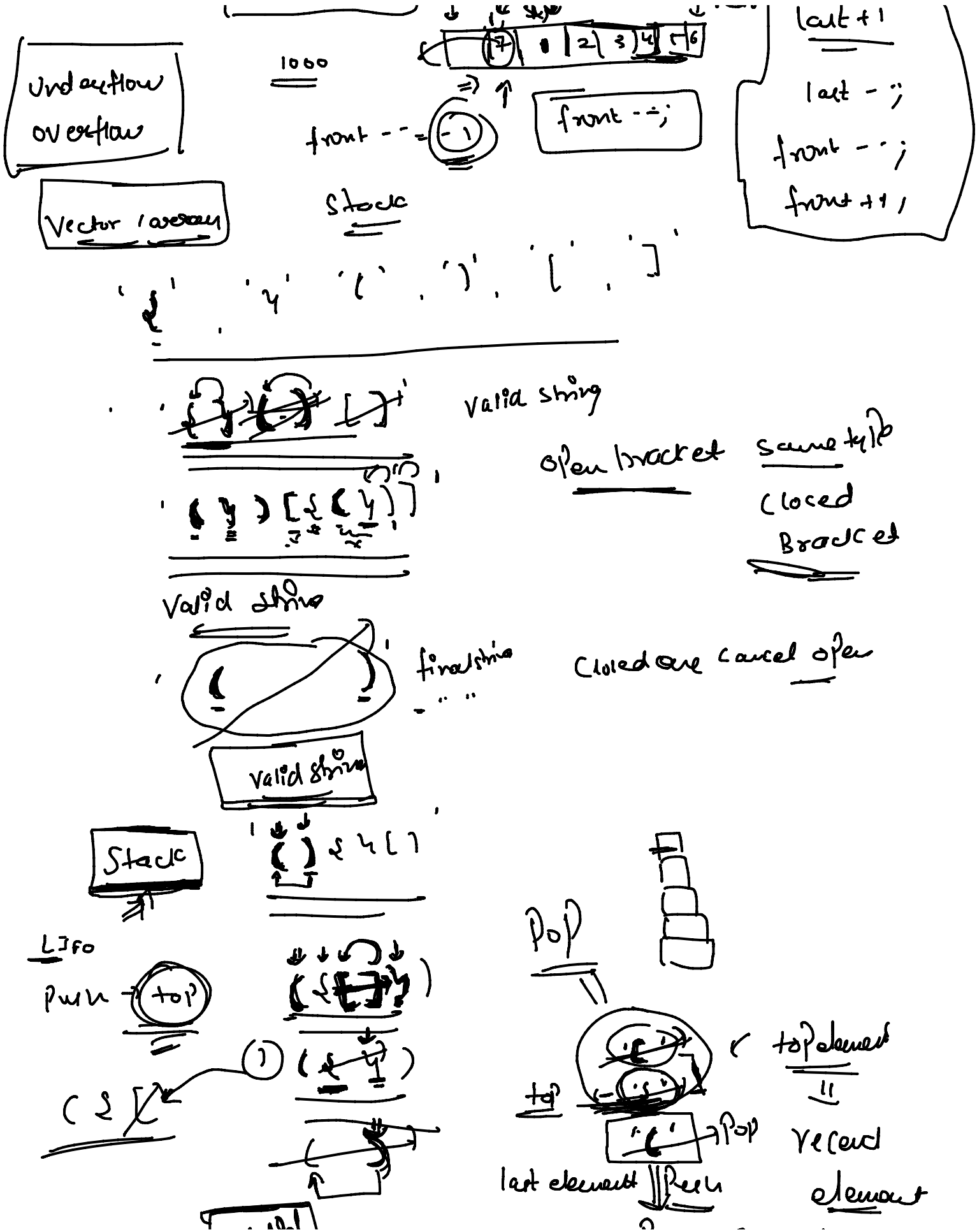
front

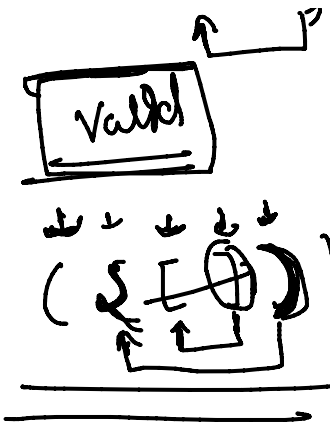
rear



last+1

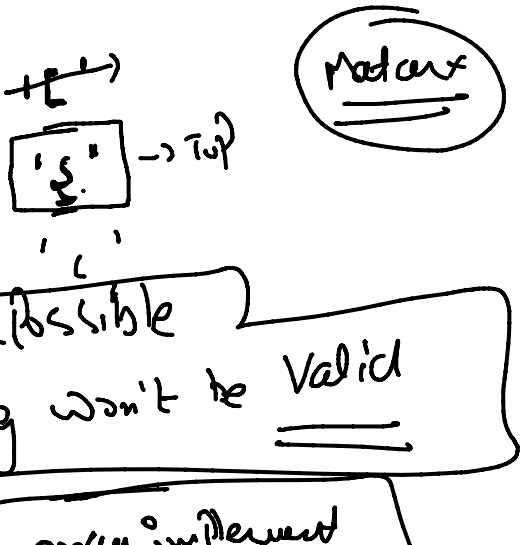
1000


Underflow
overflow
1000
front - - = -1
front - - ;
last + 1
last - ;
front - - ;
front + + ;
Vector / array
Stack
Valid string
open bracket same type
Closed Bracket
Valid string
final string
Closed are cancel open
Valid string
Stack
LIFO
push
top
pop
top element
top
pop
last element
Peek
record
element



last element ||Recent element

Recent Element

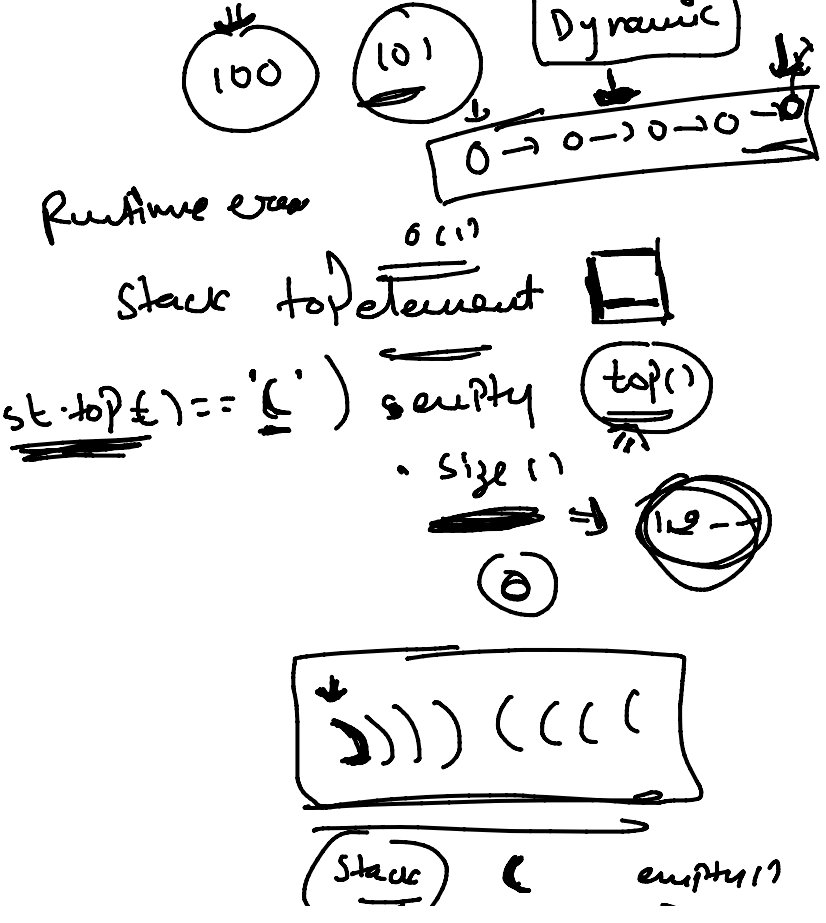


Stack

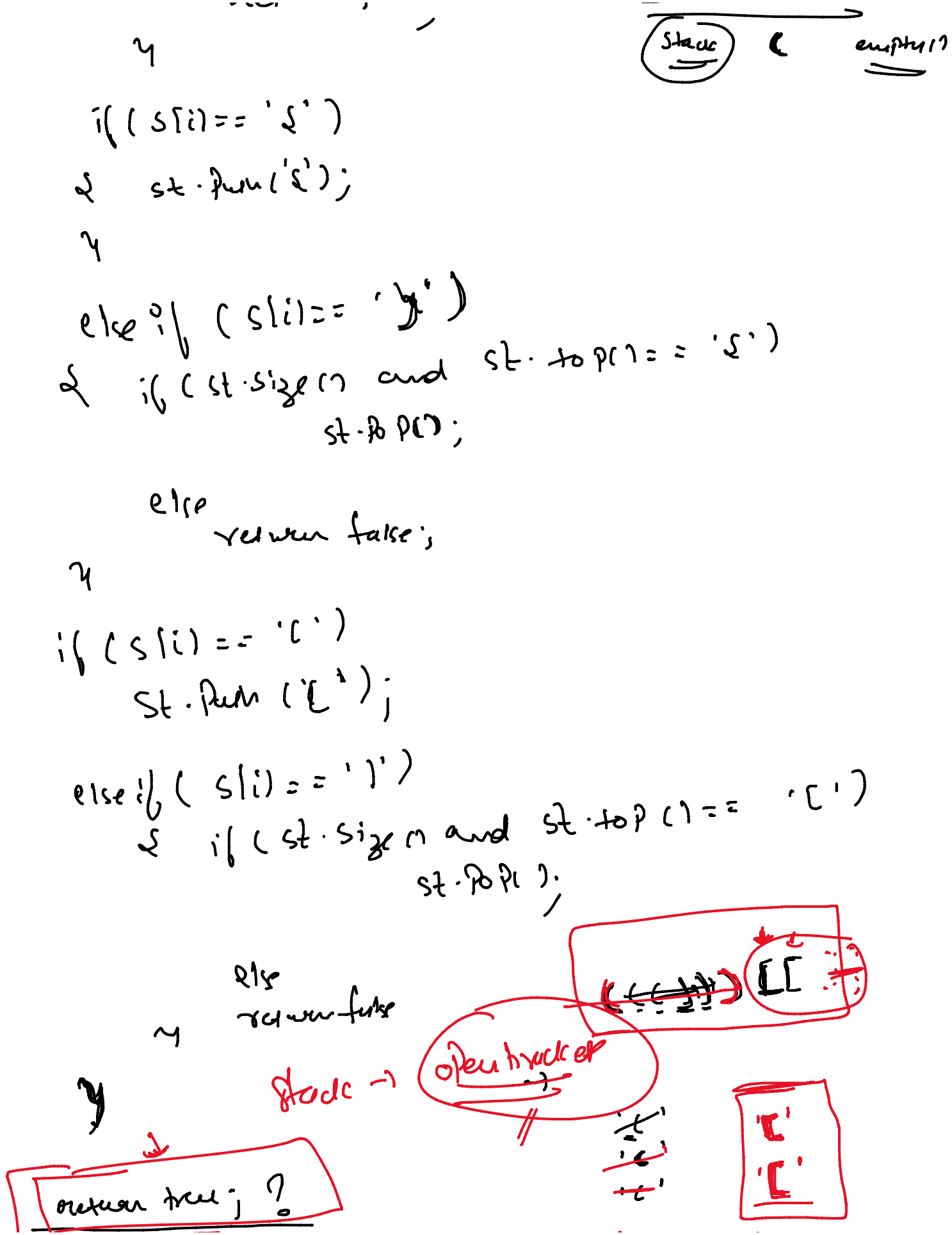
```
Stack<char> st;

int n = s.size();

For(i=0; i<n; i++)
{   if(s[i] == '(')
    {  st.push('(');
    }
    else if(s[i] == ')')
    {  if(st.size() and st.top() == '(')
       {   st.pop();
       }
       else
          return false;
}
```



```
    }
    if(s[i]=='(')
    {   st.push('(');
    }
    else if(s[i]==')')
    {   if(st.size() and st.top()=='(')
            st.pop();
        else
            return false;
    }
    if(s[i]=='[')
        st.push('[');
    else if(s[i]==']')
    {   if(st.size() and st.top()=='[')
            st.pop();
        else
            return false;
    }
}
return true;
```



return true ?

⇒

if(st.size()>0)

return false

Stack empty

↓

All characters matched

"abbaca"

abbaca ∃ ca

adjacent equal characters

5-6

final string

adjacent duplicates

abbaca ⇒ abaca

'(' ')'

flip

Stack<char> st;

st.push(s[0]);

current character

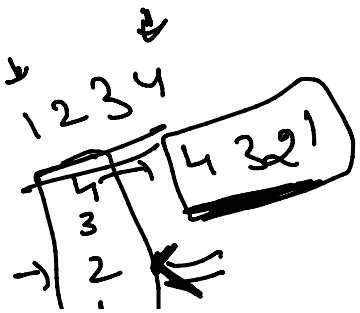
for(i=1; i<n; i++)

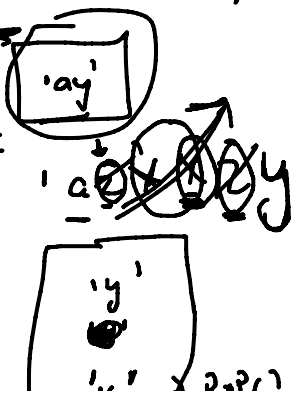
if(st.size() and st.top() == s[i])

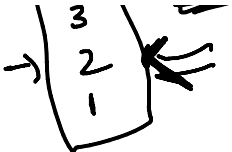
st.pop();

else

st.push(s[i]);





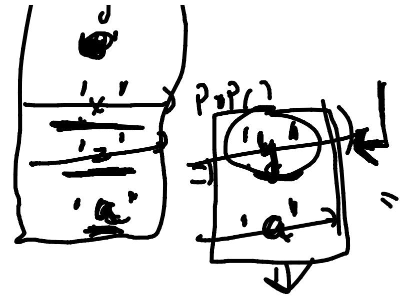


```
else
    st.push(s[i]);
}
```

reverse w/o using

```
string str = "";

while (!st.empty())
{   char ch = st.top();
    str.push_back(ch);
    st.pop();
}
reverse(str.begin(), str.end());
return str;
```



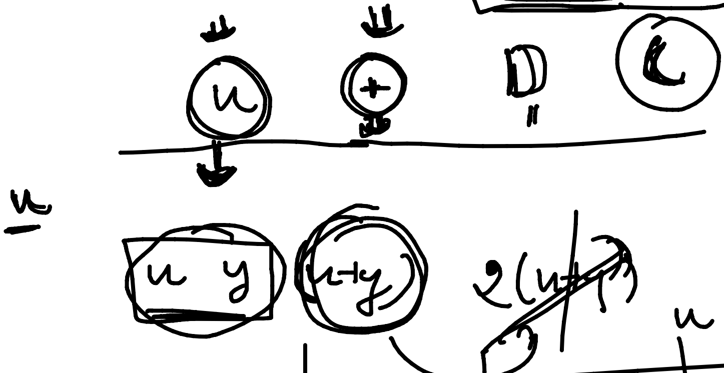
str = " "

= ya

⇓

⇒ ay

Score Card



Operation modify last value

⇓

stack

ops = ["5", "2", "C", "D", "+"]

5 ~~2~~ 10 15

⇒ 5 + 10 + 15 = 30

5 * 2 = 10

5+2 = 10

5 2 10 15

5+10 = 15

⇒ 5+10+15 = 30

```
Stack <int> score;        n = ops.size()

for (i=0; i<n; i++)
{  if ( ops[i] == "+")
   {  int temp = score.top();
      score.pop();

      int temp2 = score.top();
      score.push(temp);
      int sum = temp + temp2;
      score.push(sum);
   }

   else if (ops[i] == "D")
   {  int temp = score.top();
      score.push(2 * temp);
   }

   else if ( ops[i] == "C")
   {  score.pop();
   }

   else
```

top → stack

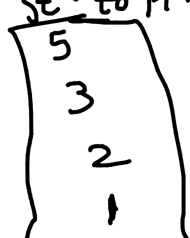
pop

st 3, 2

st.top() ⇒ 3 +

st.top() ⇒ 2

5



Topvalue double

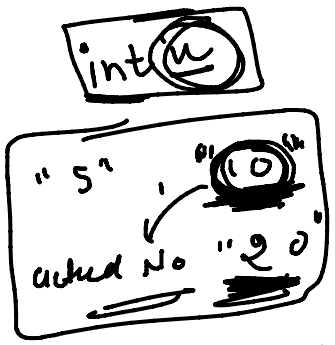
top value remove

"30345678" ⇒ int

int(n)

like

```
{ int no = stoi(ops[i])
  store.push(no);
}
}
```

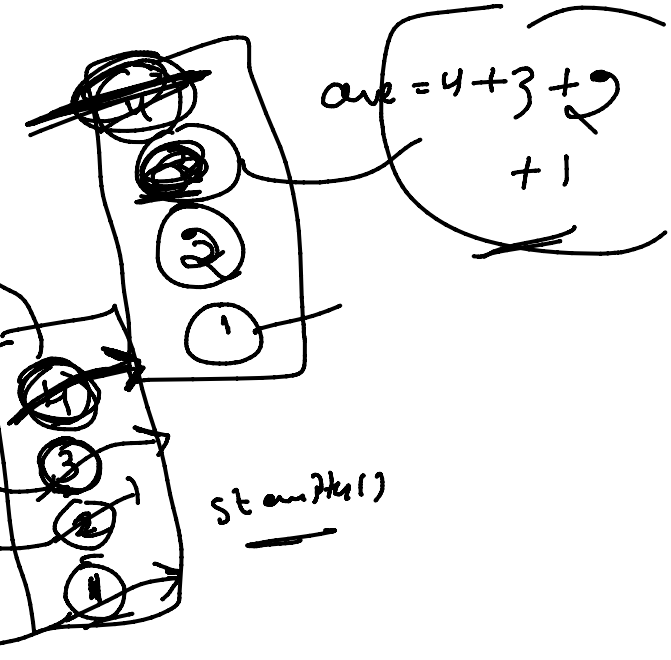


"20"

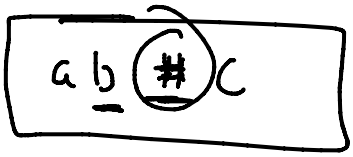
// Add all values of stack;

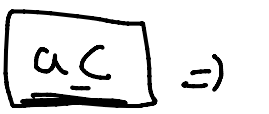
```
int ave=0;
while(!st.empty())
{  ave = st.top();
   st.pop()
}
return ave;
```

stop



BNY Mellon



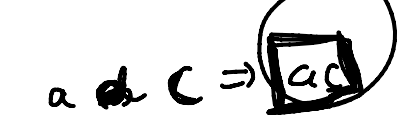


=> a c

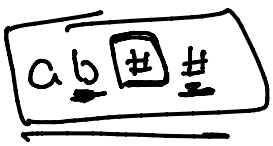
ab#c

a b c => ac

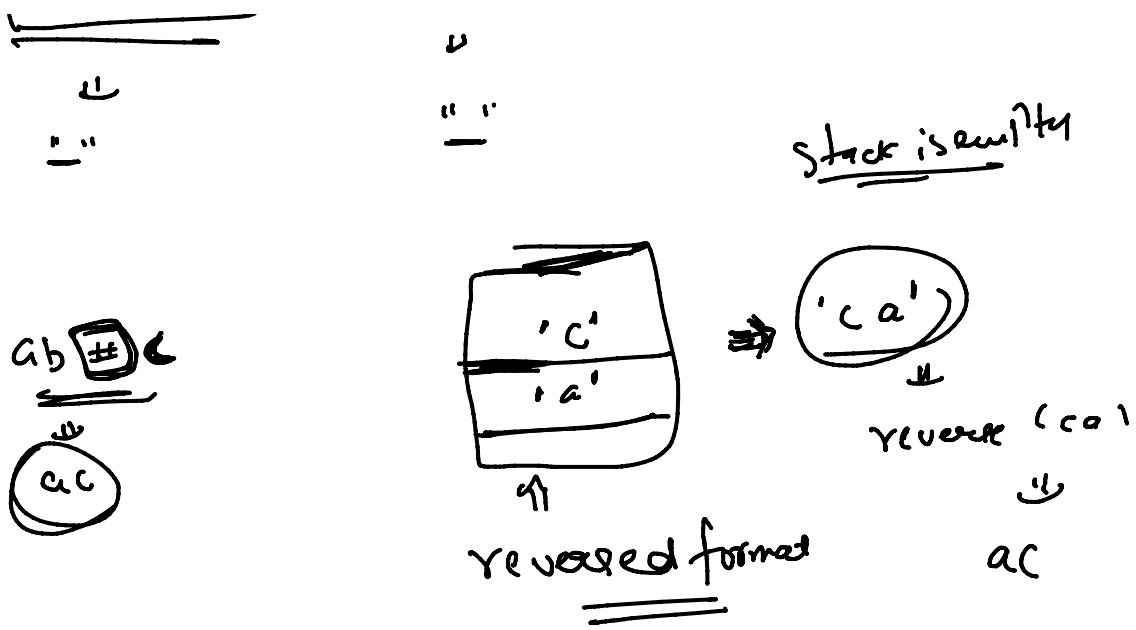
=>



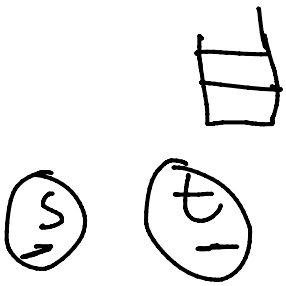
top element



a # d #




```
Stack <char> st;
For (i=0; i<n; i++)
{ if (st.empty())
    { if (s[i]=='#')
        continue;
      else
        st.push(s[i]);
    }
  else
    { if (s[i]=='#')
        st.pop();
      else
        st.push(s[i]);
    }
}
// stack final string in reverse format;
String substr="";
```



String substr = "";

while (! st.empty())
{ char ch = st.top();
substr.push_back(ch);
st.pop()
}

Stack empty()

substr.begin(), substr.begin() + 3)

0, 3

reverse (substr.begin(), substr.end()

Substr

Similarly t

Stack <char> st2;

Modules

Functions

Sprinkler;

2: 20-30

20.25

7-2