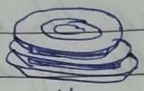
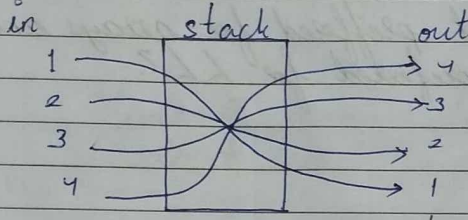
DATE

33 Stack - I

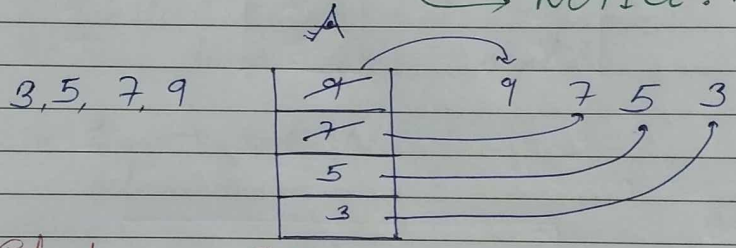
Stack → is a data structure that follows LIFO



stack of plates
Last in first out



→ NOTICE: Reversed



Stack using STL

// Creation

```
stack <int> st;

// insertion
st.push(5);
// get size
st.size()

// check if empty
st.empty()
```

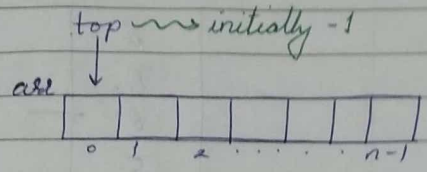
// get top element
st.top()

// delete top element
st.pop();

DATE

HW Check all functions available in stack STL

Implement Stack




```
class Stack {
  public:
   int *arr;
   int top;
   int size;

   Stack (int size) {
       arr = new int [size];
       top = -1;
       this->size = size;
   }
   void push (int ~~int~~ data) {
       top++;
       arr[top] = data;
   }
   void pop() {
      top--;
   }
   int ~~top~~ getTop() {
       return arr[top];
   }
   bool isEmpty() {
       if (top == -1)
           return false;
       else
           return true;
       }
};
```

handle stack overflow
if (top == size-1) {
cout << "OVERFLOW";
return;
}

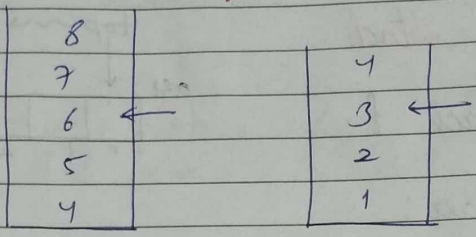
handle underflow
if (top == -1) {
cout << "UNDERFLOW";
return;
}

: top name already used

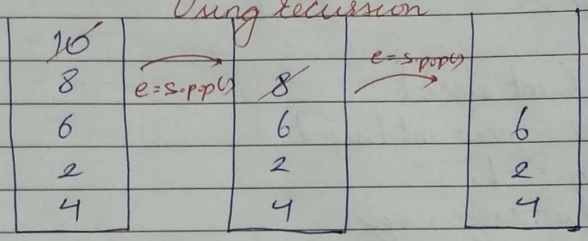
PAGE

DATE

Q Middle element of stack



Using recursion



size=5 size=4 size=3 → 6/2 → 3

BASE CASE

~~Total size / 2 + 1~~ == Size

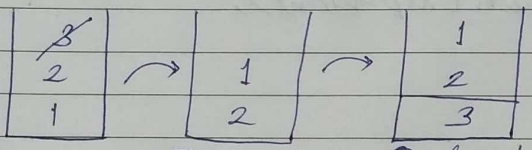
$$\frac{\text{Total size}}{2} + 1 == \text{Size}$$

```
void printMiddle (stack<int> &st, int totalSize
    if ( totalSize/2 + 1) == Size) {
        cout << "Middle is : " << st.top();
        return;
    }
```

DATE

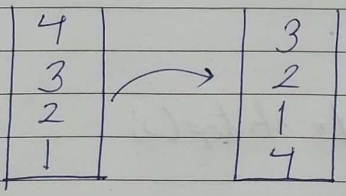
```
// step 1
int topElement = st.top();
st.pop();
printMiddle(st.totalSize);

// step 2
```

Reverse a stack

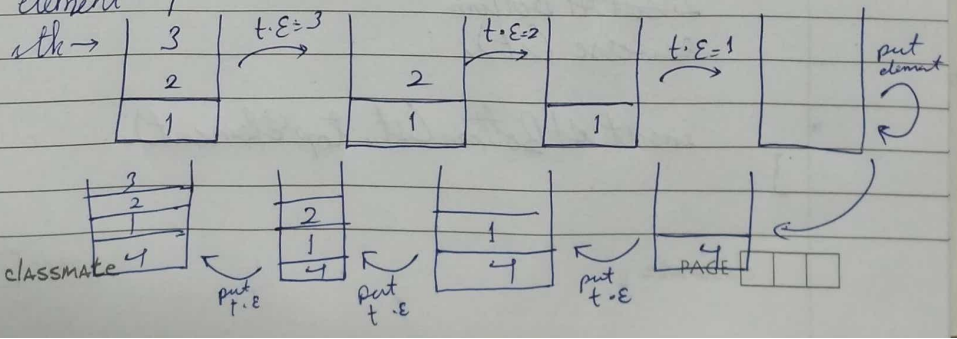


① Save top & pop ② Recursion will reverse remaining ③ Insert at bottom

Insert at bottom



element = 4



classmate

PAGE

DATE

```
void insertAtBottom(stack<int>&st, int item) {
    // base case
    if (st.empty) {
        st.push(item);
        return; }
    int topElement = st.top();
    st.pop();

    insertAtBottom(st, item);

    st.push(topElement);
}
```

Reverse

```
void reverse(stack<int>& st) {

    // base case
    if (st.empty())
        return;

    // step 1
    int topElement = st.top();
    st.pop();

    ~~insertAtBottom~~
    reverse(st);

    insertAtBottom(st, topElement);
}
```