list_t<pair_t>

vs.

list_t<pair_t*>

pair_t is just an object with a char & int as data members, e.g., (c, 4) a "tuple"

- operator you will need are operator<
operator>

```
bool operator<(const pairs_t& lhs)
{
    return (c < rhs.c);
}
```

characters determine order

- when adding to a list:

```
list_t<pairs_t> list;

list.push_back(pairs_t(45.0,
                       '5'));
```

adds a copy of paint object onto list

- push_back eventually calls

insert(iterator, const T& rhs)

possibly something like

new node_t(rhs, p->prev, p)

copy from rhs

copied

- point of this is:
use pointers of "large"
objects or lists

list_t<pass_t *> bit

pointer to object

gets copied between
push_back, insert,
etc; but only 1 long byte

- the stack

- basic ops: push, pop

(top)

peek at top

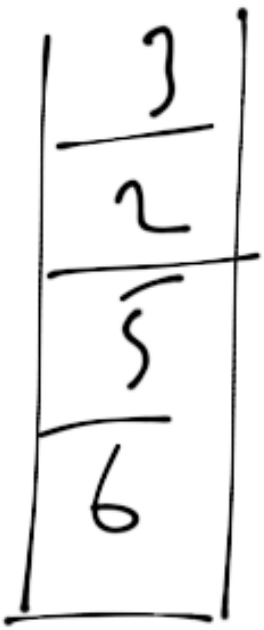
remove from top

of stack

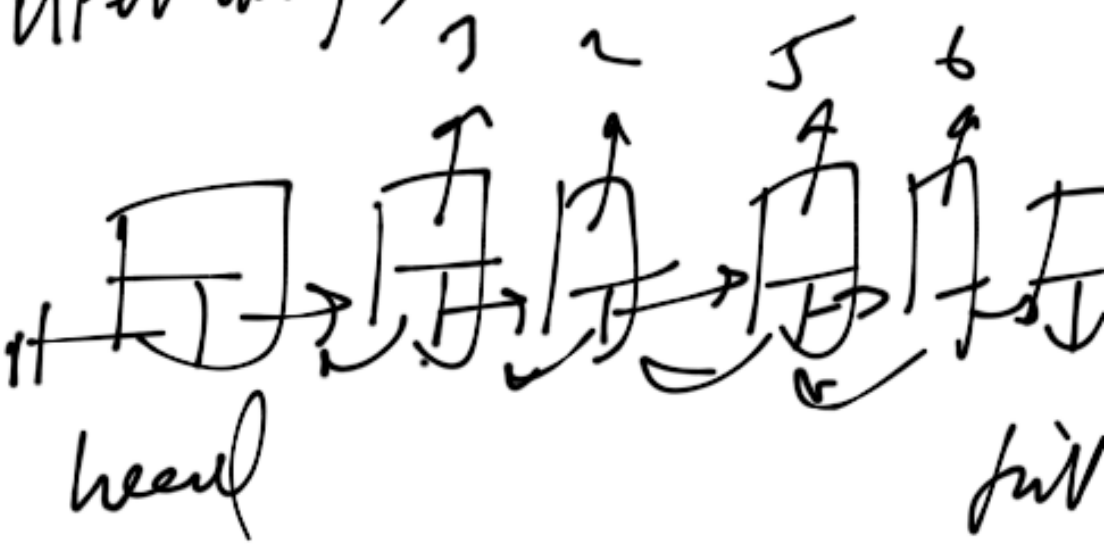
- push_front — push
pop_front — pop
front — top

- Reverse - Polish Notation

6 5 2 3 + 8 * + 3 + *

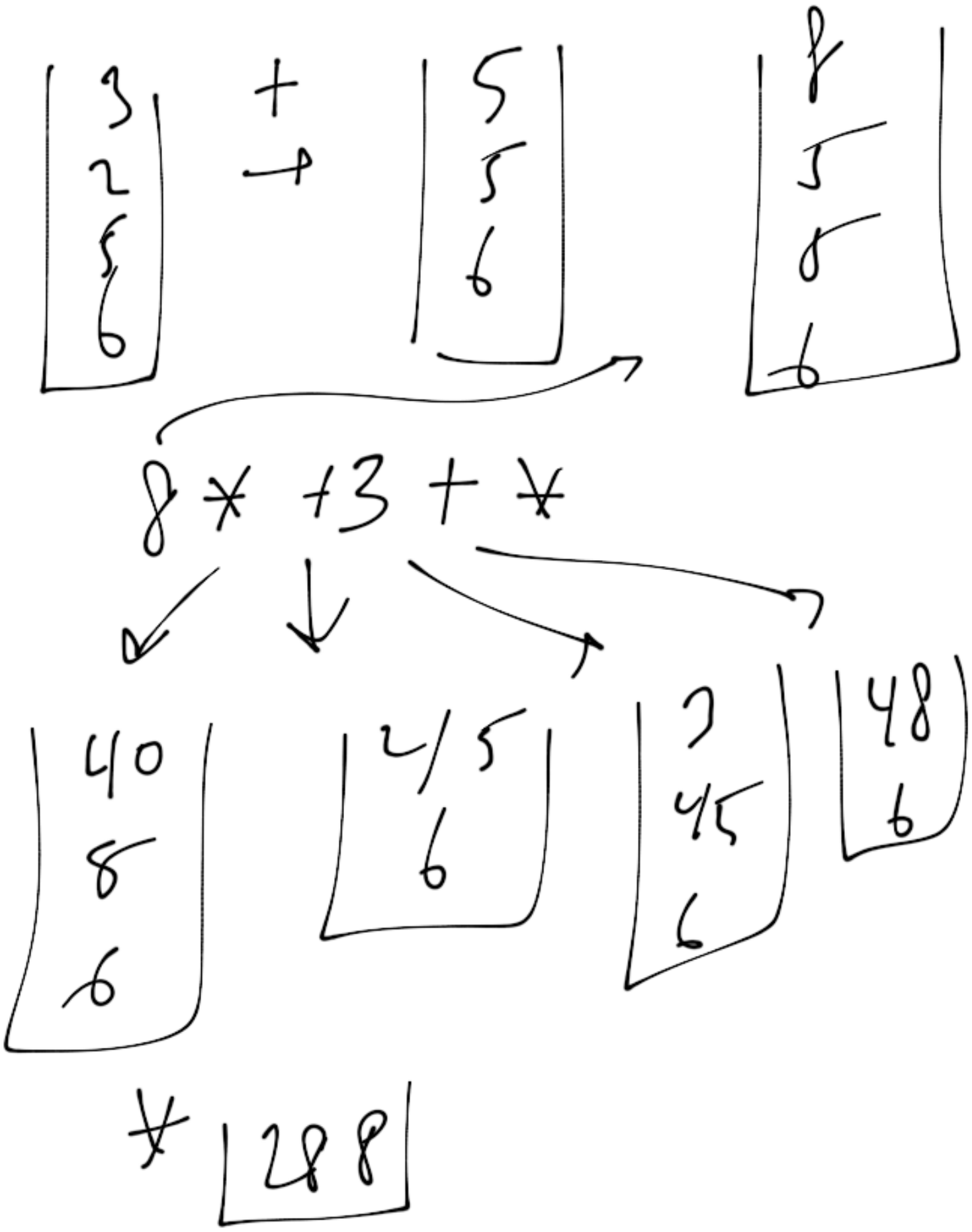


intermediate,



---

read in '+' pop off
top 2 elements, evaluate
push result onto stack


3
2
5
6
+
5
5
6
8
5
8
6
8 * 3 + *
40
8
6
25
6
7
45
6
48
6
*
288

- What about infix notation?

a + b * c + (d * e + f) * g

⇒ Convert to postfix

a b c * + d e * f + g * +

— use a stack

(for operands, use 2 stacks)

- part 1: conversion
  from infix to postfix:
- read in infix expression
  from stdin (std::cin)
- reading in characters

⇒ list_t<char>
(holds chars as we
process them)

- as we procen input,
  composing a string
  in postfix notation

    std::ostringstream os;

- os collects characters
  as they're processed

```
while(c=std::cin.get()){
```

char c

stdin

a char

```
    if (isdigit(c)) { //operand
        // convert to integer
        std::cin.putback(c);
        std::cin >> operand;
        os << operand << " ";
}
```

```
else {  // not a digit (operator)
  switch (c) {
    case '(':
      :
      break;
    case ')':
      :
      break;
    case '*':
      :
```

⋮

```
case '\n':
    // process os
    (ostringstream)
    postfix(os.str().
            c_str())
    break;
```

part 2 of lab
(evaluation of os)

```
postfix(const char * expr)
{
    list_t<int> stack
    :
    process RPN expr
    and output result
}
```

See §3.6 (p.99)