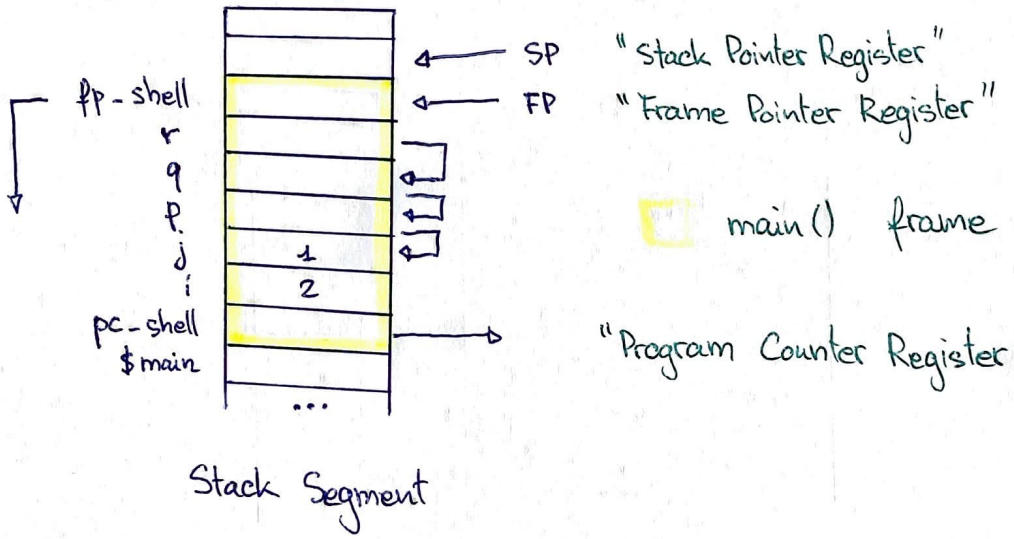
SOP - Pointers

GROUP 1
Ricci Paul
Richoz Julien

A) Pointers and Arrays + Stack and Heap Segments

①

```
int i ;
int j = 1 ;
int *p = &j ;
int **q = &p;
int ***r = &q ;
i = ***r + 1 ;
```

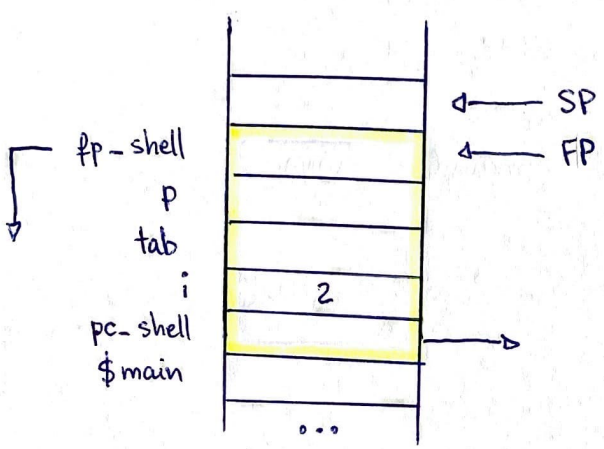


Stack Segment

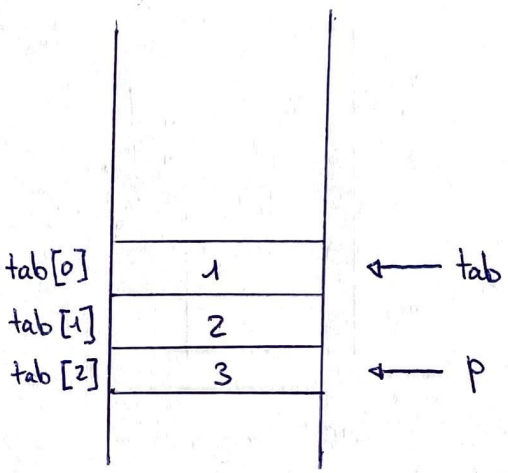
②

```
int i ;
int tab[3] ;
int *p = tab ;
++ p ;
++ p ;
i = p - tab ;          // i == (address of p) - (address of tab) = 2
tab[0] = 1 ;
(tab + 1)[0] = 2 ;
*p = 3 ;
```

"Assuming local arrays are stored in the Heap Segment"



Stack Segment



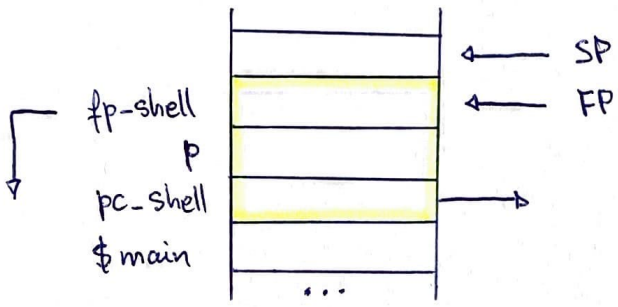
Heap Segment

main() frame

③ `int *p = malloc ( 2 * sizeof (int) );`

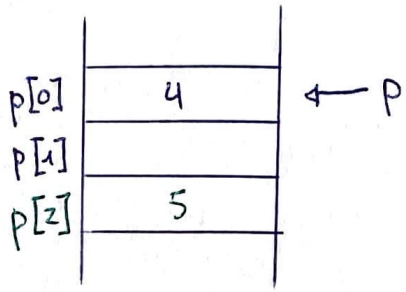
`p[0] = 4;`

`p[2] = 5;`



Stack Segment

main () frame



`p[2] = 5;`

could lead to a Segmentation Fault !

B) Pointer to Function + typedef

```
typedef int (*mathFunc_t)(int, int); // define the new type mathFunc_t
```

↳ type mathFunc_t = a pointer of int to a function that takes 2 args of int...

```
int add(int a, int b) {
    return a + b;
}

int mult(int a, int b) {
    return a * b;
}

int compute(mathFunc_t f, int a, int b) {
    return f(a, b);
}

int main() {
    mult(add(2,4), 8); // return value = 48
}
```

arg = a type mathFunc_t → a pointer to a func that takes 2 int and returns an int, like add or mult...

add(2,4) → 6; 8 → 8; mult(6, 8) → 6 * 8 = 48

```
compute(mult, compute(add, 2, 4), 8); // return value = 48
```

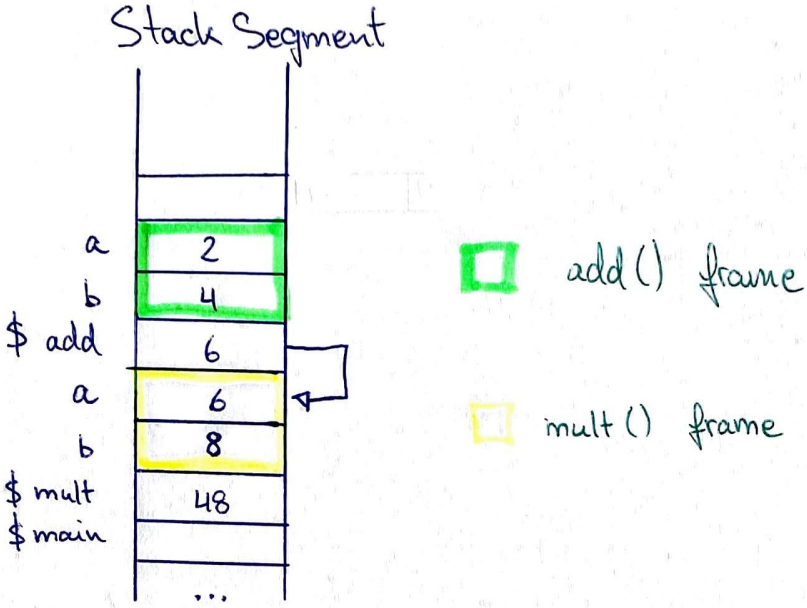
arg of type mathFunc_t

compute(add, 2, 4) → 6; 8 → 8; compute(mult, 6, 8) → 6 * 8 = 48

⇒ Simplified Stack Segments : PC → Line 1 of main()

↳ "without pc-... and fp-...

// Line 1 : mult(add(2,4), 8);



⇒ Simplified Stack Segments : PC → Line 2 of main

// Line 2: compute(mult, compute(add, 2,4), 8);