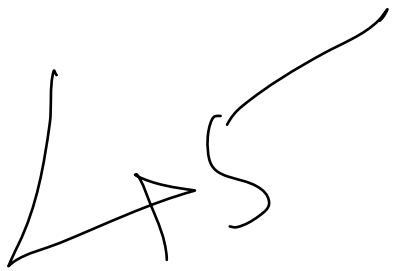
Let’s solve it



~~Passi~~ Returning multiple information back to caller (UGF)

---

(1) Group all information in single structure variable and return that variable.

i.e. name, id, m1, m2, m3

```
struct student s;
s.m1 = ;
s.m2 = ;
s.m3 = ;
strcpy (s.name, "____");
s.id = ____;
return s;
```

It does not make sense to combine un/non related things and send out.

(2) Use of pointers

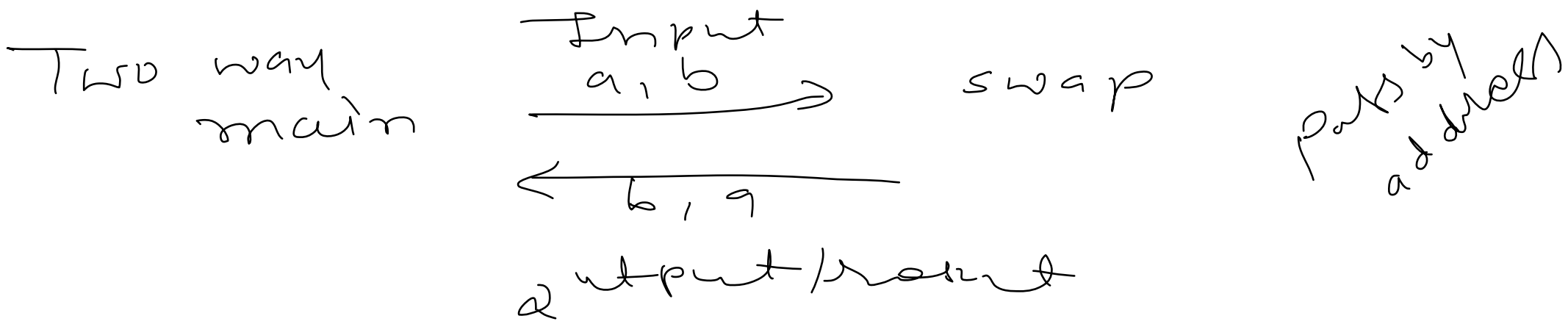
$x, y \Rightarrow x+y, x-y, x*y, x/y$

Write a function and test it using main().

In the case of exchange/swap function you already returned

the new value of a (b)

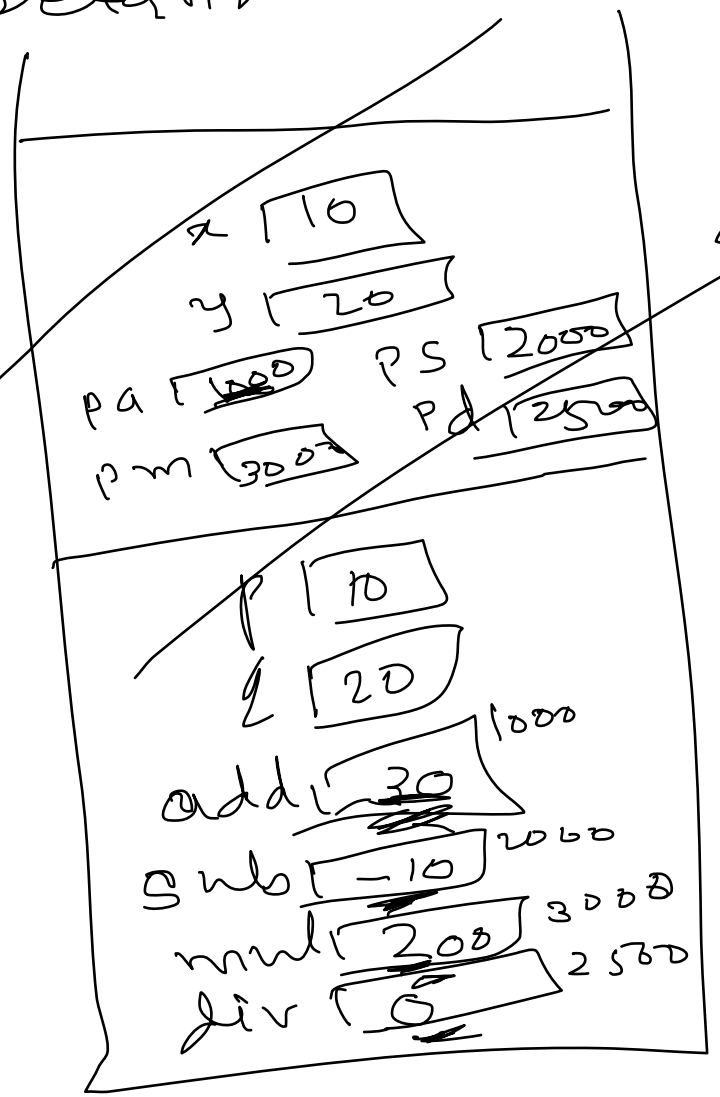
" " b (a) back.



```
domaths(int x, int y,
        int *pa, int *ps, int *pm,
        int *pd)
{
    *pa = x+y;
    *ps = x-y;
    *pm = x*y;
    *pd = x/y;
}
int main()
{   int p=10, q=20;  int add, sub, mul, div;
    domaths(p, q, &add, &sub, &mul, &div);
printf("%d", add);
```

main ⇄ domaths

return



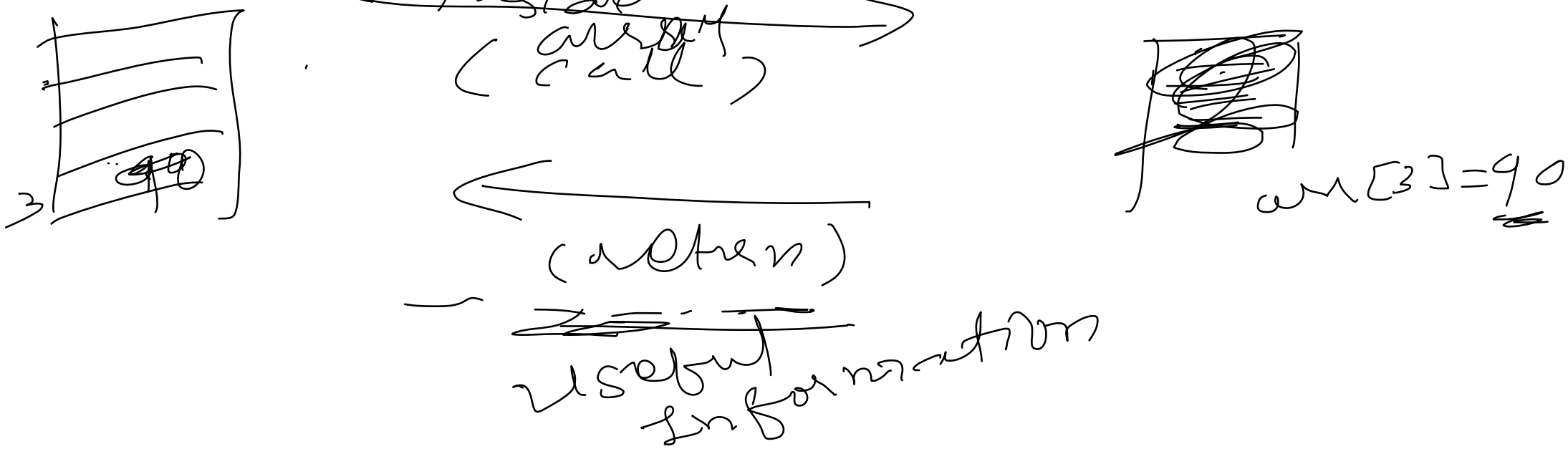
domaths

main

```
*pa = x + y;
*ps = x - y;
*pm = x * y;
=
```

Notice that domaths function has indirectly returned four information back to caller main
- additi
- sub
- mul
- div

③ You can use dummy array as an argument



```
void domaths(int x, int y, int l, int []ans)
{
    ans[0] = x+y;
    ans[1] = x-y;
           = x*y;
    ans[3] = x/y;
}
```

```
main()
{
    int array[4];
    domaths(p, q, array);
    for(i=0; i<4; i++)
        printf("%d", array[i]);
}
```

p=10, q=20;

// garbage initially.

```
struct AllInOne
{ int a,s,m,d;
};
struct AIO domaths (int x, int y)
{
    struct AllInOne aio;
    aio.a = x+y;
    aio.s = x-y;
    aio.m = x*y;
    aio.d = x/y;
    return (aio);
}
int main()
{
    int p=10, q=20;
    struct AllInOne res;
    res = domaths(p, q);
    printf("%d", res.a);
}
```

Pointer to array

If i have pointer to single integer

if I do pointer + 1

It jumps 4 bytes (one single integer)

Pointer to array

(array pointer +1)

will jump whole array.

pointer to a row

it should jump row

int arr1d[10] = { }

int *ptr = arr1d ;// pointer to single int

Not just a pointer to array

only in 2D or more

int arr2d[5][10] ; // pointer to a row containing 10 (colums) elements

int ( ) = arr2d ;

int [10]

ptr+1 ⇒ next row beginning.

Type 2

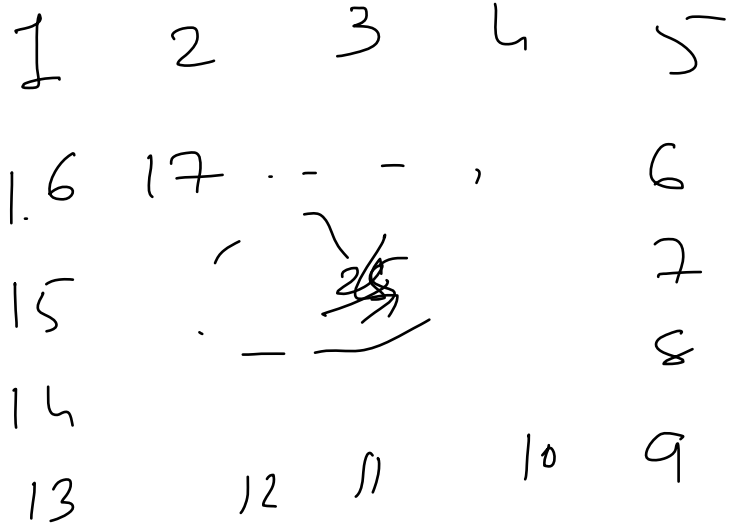
10

int arr[10] ;

int (*ptr)[10] ;

ptr = arr ;

Type 1


1 2 3 4 5
16 17 6
15 7
8
14
13 12 11 10 9

pointers and structures

— access member of a structure variable via pointer

```
struct student s = {1, "John", 10, 10, 20};
struct student *ps;

ps = &s;

s.id   // (*ps).id   // ps -> id
```

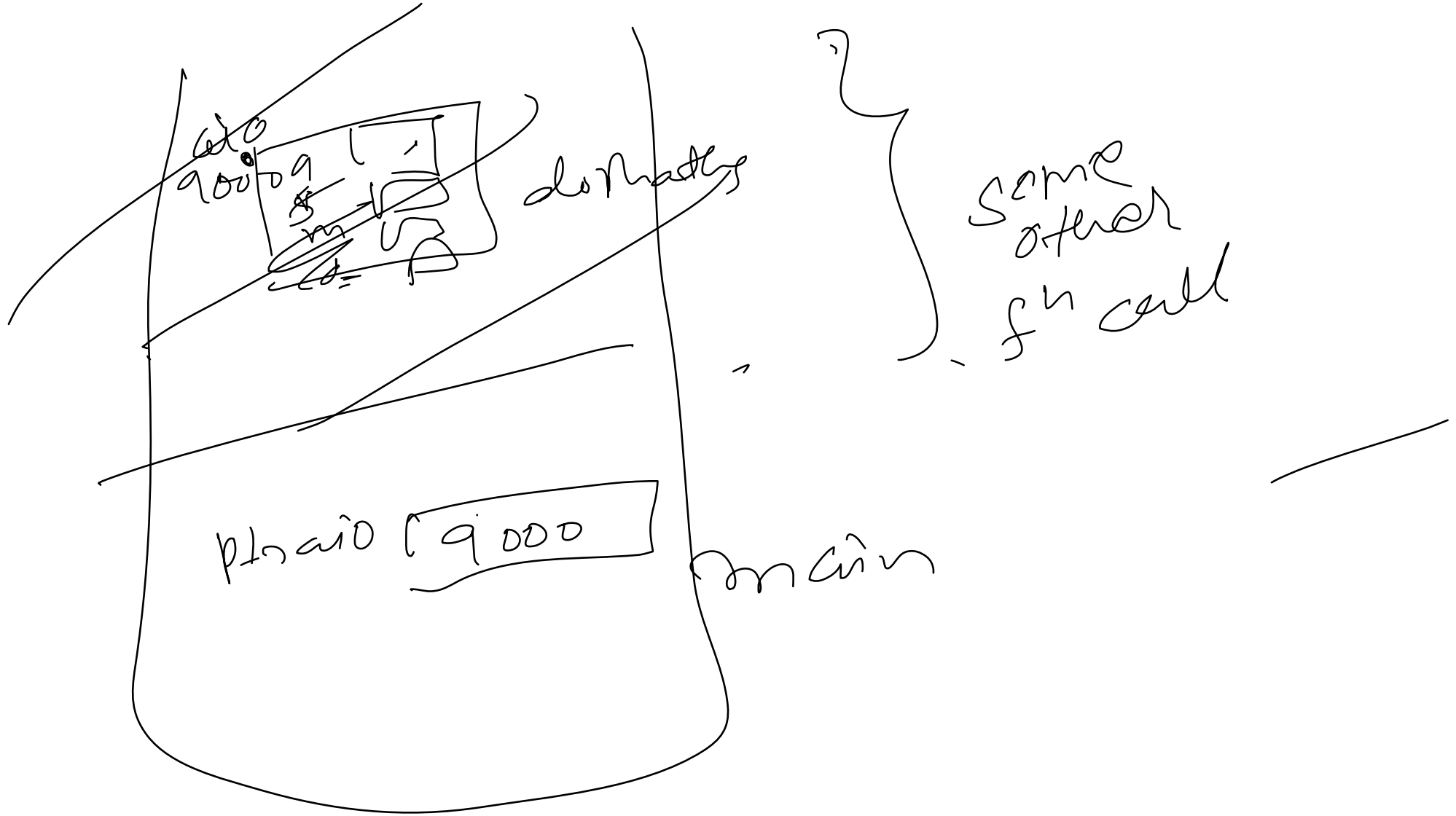
Arrow operator

Pretty Problem

```
struct Alliance *domaths( int x, int y)
{
    struct Alliance aio;
    aio.a = x+y;
    aio.s = x-y;
    :
    return &aio;
}
```

Do Not point a return address of local variable. Otherwise syntactically correct.

```
int main()
{
    int p=10, q=20;
    struct Alliance *ptraio;
    ptraio = domaths(p,q);
    printf("%d, %d, %d, %d",
        ptraio->a,
        ->s
        ->m
        ->d);
}
```


do
doMaths
same other fn call
ptr a10 9000
main