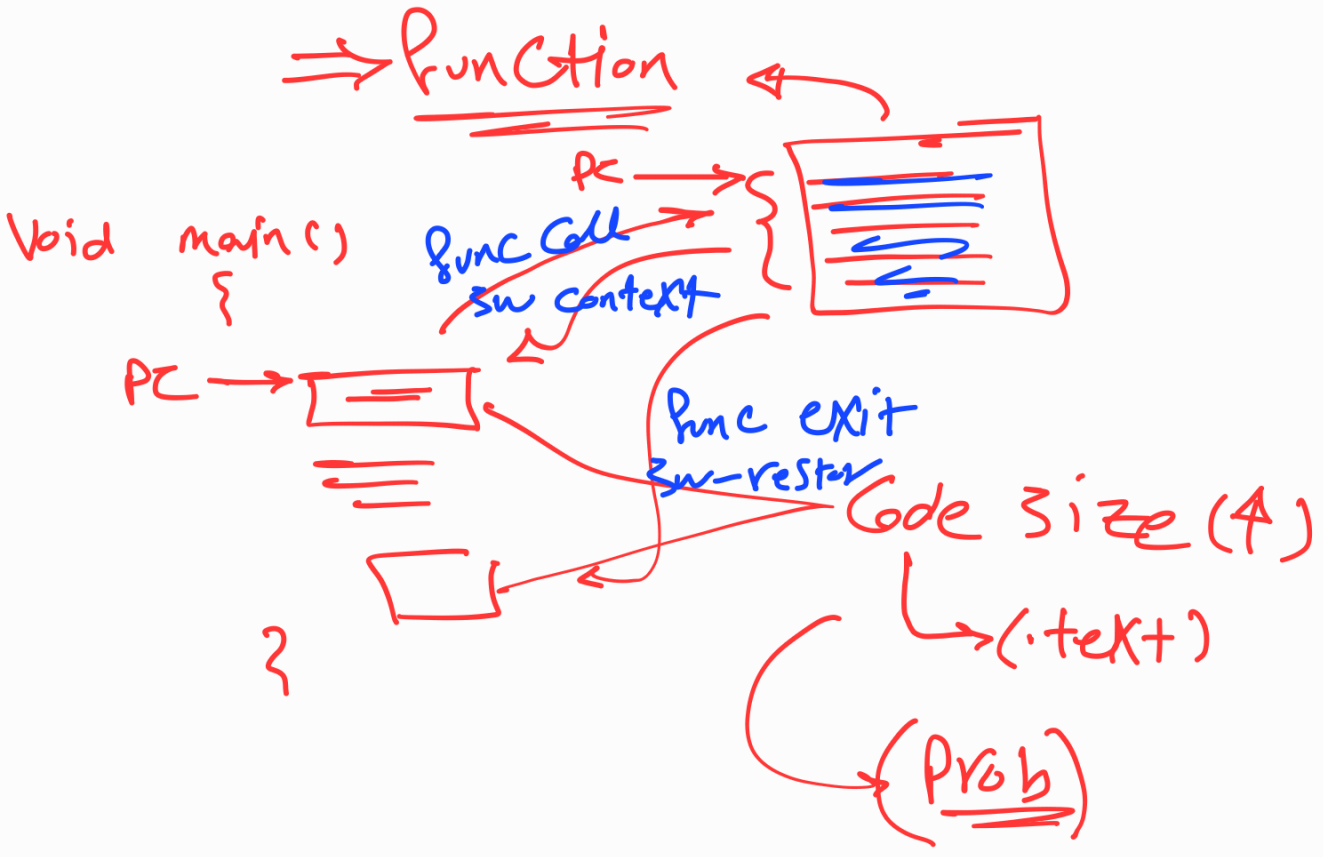


Sol:- ⇒ func

mem → Adv

time → dis-Adv

Adv

- less Code size
- organize Code

dis-Adv

- execu time (↓)

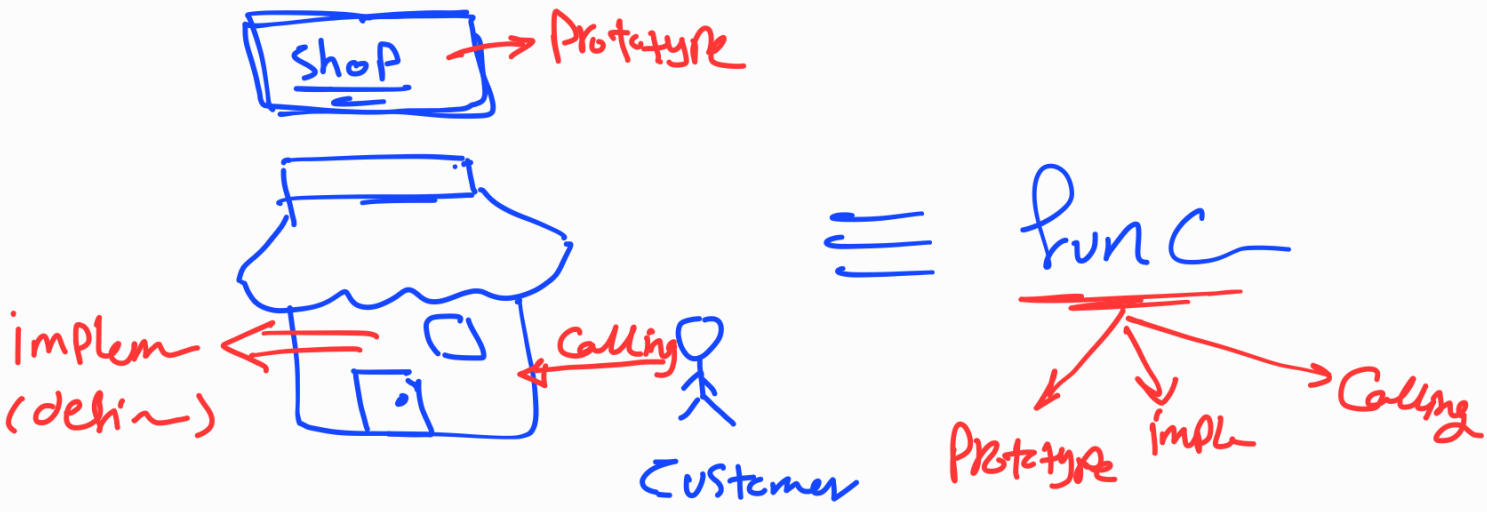
Memory vs time

Memory ✓ X

time X ✓

- what?
- why?
- How?

↳ example



[1] func- Prototype (declaration)

return_type func_name (argument list);

Eg

int sum (int x , int y) ;

return type → doesn't return
↳ (void)

default ⟹ int

int-func () {

(garbage value)

ret- x

}

Hints

⟹ Argum— ⟶ No ⟹ Void

⟹ Multiple def ⟹ error

⟹ No Prototype

Compiler

- ↓ generate warning & implicit decla—
- ↓ generate error & Stop Comp—

---

2 func implem— (definition)

syntax

```
return_type func_name ( arg— )
{
    ——— code
    ———
    ———
    ———
    return [ ] ;
```

⟹ return Stat—

```
    ———
    ———
}
```

⟹ un readable code

un reachable code

```
if ( ) {  ≡
    return — ;
}
else {  ≡  }
```

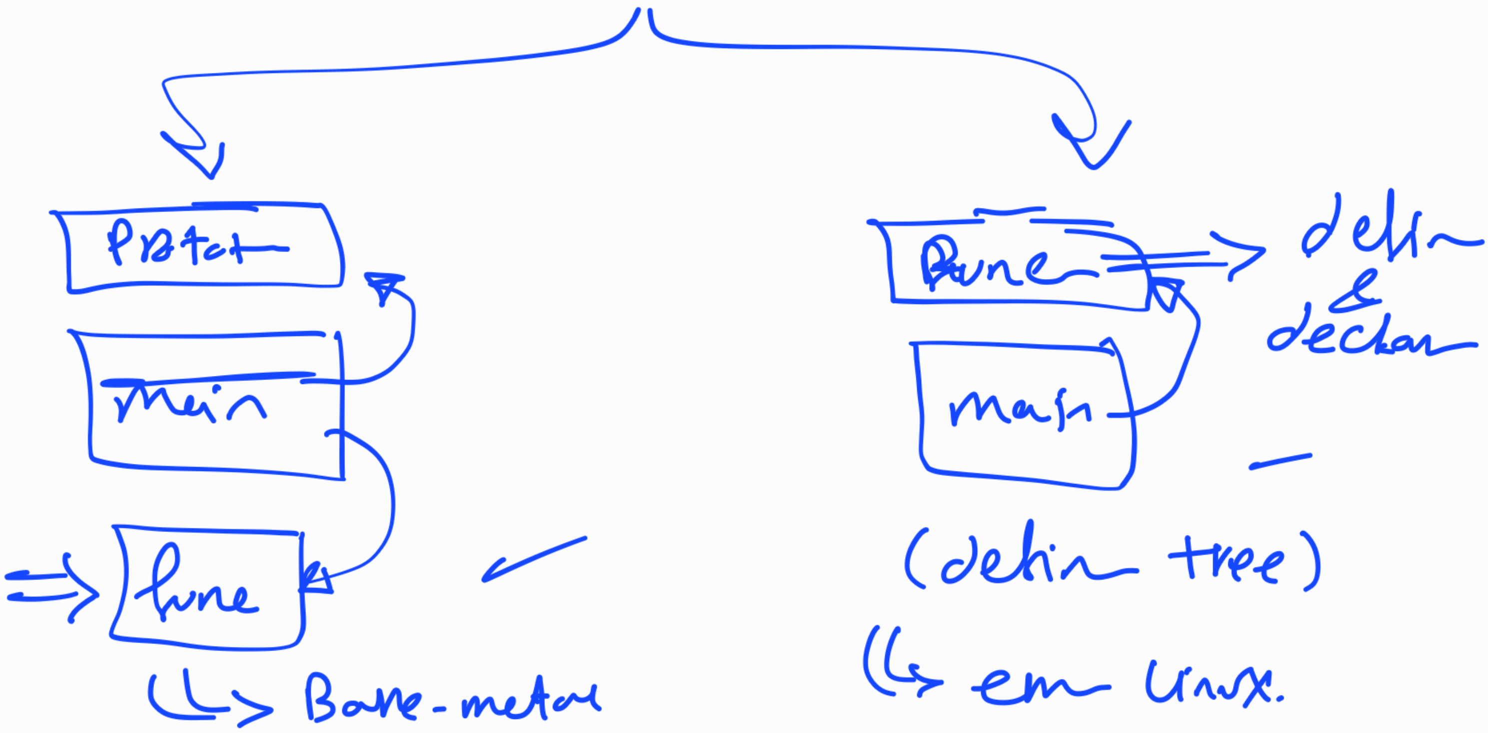
Hint 3

(*) func impl → inside another func (X)

ret fun (—) ⇒ prototype

main ( )
{
int x;
Calling ⇒ func(); ⇒ Calling
}

ret → func ( ) ⇒ defin
{
}



[3] Calling

↳ func Name → Void
func ( ) ; → X

* main → Calling ?
↳ Startup Code

---

* func errors :-

① return (X) ⟹ garbage

② Void fun ⟹ retu ?
↳ Compil error

③ return name (int X, Y) ⟹

int x , int y

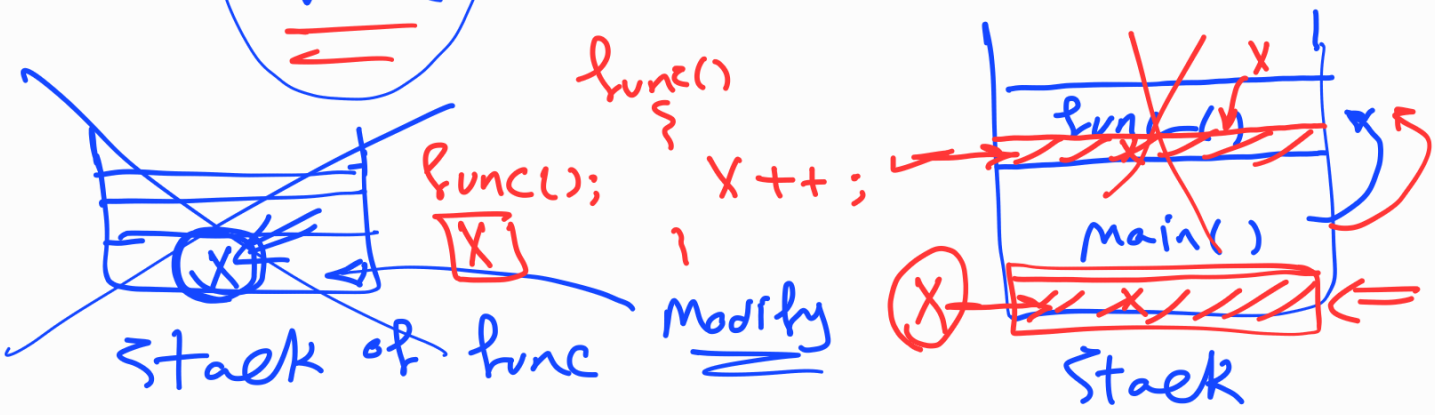
④ return name (~~X~~) ;

{
int x → redeh
}

⑤ Parame → redefin local var ⟹ Compi- error

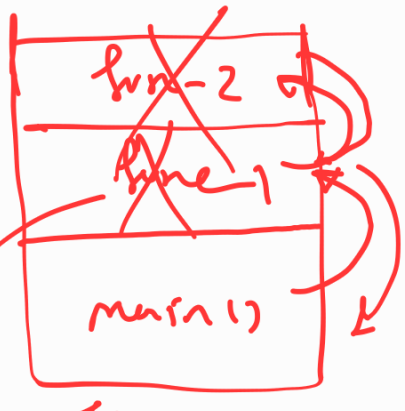
⇒ Passing the Argu

- Pass by Value
- Pass by Ref



↳ Pass by Value

↳ Argum (Vary



① Copy ⟶ Nomodify

② memory size ③ time

(Not Secure)

Sol ⟹ Call by ref

- Addr

hint

scanf("%i", &X);

Calling

- Value → Variable
- Ref → - Arr, - Point, - String