ELF : executable & linking format

readelf ● -- segments < >
-- sections < >

Pointers (*)
(&)

$\overset{type}{\downarrow}$ int $\overset{name}{\downarrow}$ *ptr

address of operater

(address of an int)

N
EAP

Email instructor to find where you lost points on lab Ø1

- min 6 test cases in script file
- code as if user is an idiot [Lab Ø2 check value of n]
- repeated everything in Lab Ø1 email

Pointers:

```
int number;
int *ptr;        // same as int* ptr and int * ptr
ptr = &number;
*ptr = 5;        address of de-referencing operater
```

<u>Pointer</u>

```
int *ptr = &number
*ptr = 5
ptr++        [next address]
[should initilize]
```

<u>Reference Vars</u> - used for overloaded functions

```
int &rVar=number;
rVar=5        [implicit derefrencing]
[cannot access directly]
[must initilize]
```

Profiling, Code coverage, memory leaks

time check each f

Header formating

```
#5 . . .
# . . .
int square(int)                  → prototype
int main() {                        → actual param
    int res = square(5)
  }                                 → formal param
int square(int num) {
     return num *num;
   }
```

		Pointer	Ref Value
	Call by Value	Call by Ref	Call by Ref
prototype	int square(int) ↓	void square(int *) ↓	void square(int &) ↓
call	square(num)	square(&num)	square(num)
	function copies num into new address to be used in function	same address	same addres "safer" than pointer
return statement	num *= num	*num *= *num	num *= num

Figure out how "const" work with above