Exam 3 Review

Literals

- Identify literals in the below code:

```
1. char *arr="Yellow";
2. char str[]="Yellow";//str[]={‘Y’,’e’,’l’….}
3. int x=100;
4. int y=0x1234;
5. int z=0xDeadbeef;
6. int a=0xafee;
7. int b=0xlol;
```

Types

- What are the types of the following?

```
1. int x, *y; //type of x, &x, *&y, *&x
2. day[1]=“Monday” //type of day, day[1], &day, &day[1]?
3. int foo(double, char* c) //type of foo
4. char* (*)[7] //what does this type mean?
```

When does array name not behave like a pointer?

1. In multi-dimensional array scenario
2. With sizeof
3. With & (Address-of) operator

```
char* curday[1]={"Tuesday"};
char day[7][15]={"Mon","Tue","Wed","Thu","Fri", "Sat", "Sun"};
int intarr[2]={0,1};

//day[0]="Friday"; //exception 1
printf("%zu -- %zu\n",sizeof(curday), sizeof(day)); //exception
2
printf("&intarr:%p intarr:%p &intarr[0]:%p\n",&intarr, intarr,
&intarr[0]);      //exception 3
```

Const

- Read from right to left

```
int x=10;
const int *p=&x;//p is a pointer to constant integer
int const *p=&x;//p is a pointer to integer constant
int *const p=&x;//p is a constant pointer to an integer
```

*Remember it doesn't matter where you place the * in between a blankspace.*

Const – compile-time and runtime errors

- Compile-time errors – assigning values to some variable whose type is known to be a const (by the compiler)
- Runtime error – assigning values to a variable which is in RO memory (e.g. literals)

```
int x=10;
const int *p=&x;
*p=20; //compile time error
p[0]=20; //compile time error

char* str="Hello";
str[0]='Y'; //runtime error.
```

Concluding Remarks

- Understand the requirements/spec
  - Look for minute details
  - Practice reading the spec/requirements doc
- Think of all possible error scenarios
- Do not make assumptions
  - Ask if the spec does not state something about the assumption

Thank you