**Making Decisions:**

1. when we write:
2. if((score >= 90) grade = 'a';
3. else if (score >= 80) grade = 'b';
4. else
5. grade = 'f';
6. printf("%d \n", grade);
7. return 0;
8. there is only one possible solution
9. during if else statements,
10. if(x<y)
11. if(x>y)
12. else
13. x++
14. will run one if statements and the else statement
15. indentation DOES NOT matter in C, similar to java we must do proper formatting, not like python

- we use if, else if, and else to provide options based on certain situations

- it will skip past the else once a condition is satisfied

- certain formatting toold like {} will allow us to properly organize this code

**nested conditions:**

- it is possible to create conditions that are nested into eachother, but also we could rewrite them without requiring nested conditions (all in one line)

**Testing:**

- it doesn't matter how MANY unique tests you do, but it matter the quality

- getting each range of testing, hitting boundaries, and unique tests for each (5 in example on slide 9/19)

1. if(x < 10){
2. if (x > 5) printf("Small");
3. else printf("Very Small");
4. else{
5. printf("Very big");
6. return 0;
7. this code requires 5 cases, for <5, 5, 5<x<10, 10, >10
8. int i = 3;
9. if (i %2 == 0)
10. if( i == 0) printf("zero\n");
11. else printf ("How odd\n");
12. this case would print nothing because the first statement is false, which means that all three would be skipped due to the lack of curly brackets around the second if statement

**Damaging else rule:**

- when each else statement is not properly assigned to the if above, causing the code to return the wrong valur

**Ternary conditional operator:**

expr1 ? exper2 : expr3

- if e1 is true, then e2 will be **returned**, but otherwise it will **return** e3

**eg)**

printf("%d", i > j ? i : j);

returns then prints the maximum of i and j (the higher number)

**Switch Statments:**

1. switch(e){
2. case const\_e : statements break;
3. ...
4. case const\_e : statements break;
5. default : statements;
6. }

- in C, the break command lets us know that the case is finished

- you are not permitted to have multiple equal cases

- If no break appears, then it will flow down to the default statement

1. switch(e){
2. case 1:
3. printf("your number is 1");
4. break;
5. case 2 : ("your number is 2");
6. break;
7. default:printf("invalid");
8. Without breaks, the program will run the case it satisfies but ALSO will execute the next cases until there is a break, so be careful
9. case 1: case 2: case 3: statements; break;
10. case 4: statements; break;
11. default: printf("illegal"); break;
12. if e is case 1, 2, or 3, then it will run first statements, if case 4, then second statement
13. if(-5)
14. {
15. printf("a");
16. printf("b");
17. }
18. else
19. printf("c");
20. printf("d");
21. will just print abd, because -5 is true and the c is tied to the else statement, but not the d, so d is still printed