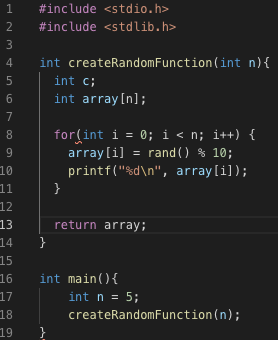
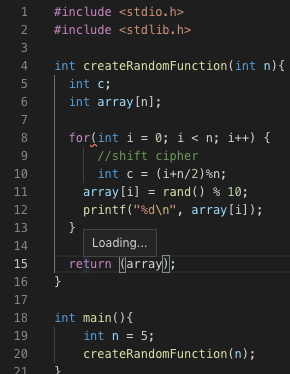
5020

Hw2\_ungraded

1. Check if equation is invertible
   1. This is not an Invertible function due to not being 1 to 1 and an onto function. Both domain and codomain have 1 element that have not been mapped.
   2. Not an invertible function because the codomain has 1 element that is not mapped.
   3. Function is invertible. Passes the 1 to 1 test and is an onto function
2. The function y = 2x mod 5 is invertible. Each element maps to another element that is both random and not already used making it a permutation function. {x,y} = {0,0},{1,2},{2,4},{3,1},{4,3}.
3. Function Signatures
   1. Using equation |b|^|a| we can see the answer is 5^4.
   2. Answer is similar to 3a. b^a
   3. There is no signature because the function is not invertible.
   4. Because permutation function codomain does not repeat, the equation would be 4!
   5. If a = b then a or b amount of permutation function exists.
4. Random Function



1. Using shift cipher to get random numbers



1. Element Probability
   1. The chance of 0 being in element 0 is 1/b
   2. Chances of f(x) =1 given f(x) is still 1/b
   3. Chances of f(1) = 0 given f(0) is 1/b. Reason being the function is not invertible
   4. Because we are working with a permutation function, we know that the function is 1 to 1 and onto. Therefore the chance of f(0) = 0 is 1/b
   5. Probability if f(x) = 1 given f(0) = 0 is 1/(b-1)
   6. Zero is already assigned to f(0) so there is a 0% chance
2. Bits
   1. A char = 1 byte = 8bits. There are 2^8 possible char values = 256.
   2. Similar to 7a, the answer is 2^64. There is 2^64 possible strings
3. If (x <= 8 | x >= 2)

printf(“guess standard deck”)

else

printf(“guess pinochle deck”)

If given 1 activation, if the card was to fall between 2 and 8, guess would be 100% correct. Else if the card was 9 or above assuming its a standard deck. The probability that I am correct is 6/13. 1 - 6/13 = 7/13 chance I am correct

1. //q = is number of times we can role.

//x = the number shown on dice

Int q = 0;

while(q != 34){

if(x >= 30){

printf( “guess 34 sided die”);

q=34;

}else

printf( “guess 30 sided die”);

q++;

}

Problem is this loop will run forever if we can roll unlimited times so I set the maximum to 34. That way we got 4/34 chance of rolling a number greater than 30 and exit the loop assuming if the die is the 34 sided one. If in all 34 loop there are no numbers rolled greater than 30, then we can safely assume we are using a 30 sided die. Running more test cases maximizes probability of success finding which die is being used.

Similar to the last problem if the die rolls 31 or above, we got a 100% chance its a 34 sided die. Any other number the probability of being the 30 sided die is 30/34 -----> (1 - 30/34)^q