Qusetion No.1

From online lecture 7 : Lemma 1: THEOREM There is no program A(P) that returns  
T, if P is a correct Python program that halts on all inputs. F, otherwise.

From online lecture 7 : Lemma 2: There is a Python program that takes an ASCII string P as input and outputs  
T, if P is a syntactically correct Python program and  
F, if P is not a syntactically correct Python program.

Lemma 3: There is no program A(P) that returns

T, if P is a correct Python program that never halts on all inputs. F, otherwise.

Proof: To obtain a contradiction, suppose there is such a program A.

def S(P):  
if P is a syntactically correct Python program:

return True

else:

return False

def HH(P, x):  
Q = "s = " + x + "\n" + P

if S(P):

return False

else:

return NOT(A(Q))

(where s is the input of Q, Q is a syntactically correct Python program if and only if P is a syntactically correct Python program.)

Proof:  
If P is syntactically incorrect, then H(P,x) = F. Since Q is syntactically incorrect, HH(P,x) = T.

If P is syntactically correct and halts on input x, then H(P,x) = T. Thus Q is syntactically correct and halts on all inputs,  
so A(Q) = T and HH(P,x) = F.

If P is syntactically correct and doesn’t halt on input x, then H(P,x) = F.  
Thus Q is syntactically correct and doesn’t halt on any input, so A(Q) = F and HH(P,x) = T.

So H(P,x) = NOT(HH(P,x)). HH solves the halting problem. But halting problem is not solvable, so HH(P,x) does not exist.

Proof for Question No.1:

Suppose there exist a function C(Q), that, given input Q, outputs:

T, if Q is a syntactically correct Python program, there is some input on which Q halts, and there is some input on which Q does not halt.

F, otherwise.

def HHH(M):

if C(HHH):

while True:

pass

elif M == “1”:

while True:

pass

elif M == “2”:

return 240

If C(HHH) = T: For HHH(M), whatever M is, HHH(M) won’t halt. So, C(HHH) should be F.

If C(HHH) = F: For HHH(M), If M = “2”, HHH(M) halts. If M == “1”, HHH(M) won’t halt. There is some input on which Q halts, and there is some input on which Q does not halt. So, C(HHH) should be T.

Then this HHH(M) is a contradiction. Thus this kind of C(Q) does not exist.

Proof for Question No.2:

Base case: when there is only one square, the length of the periphery of the resulting shape was 4, an even number.

Suppose when there are k squares, the length of the periphery of the resulting shape is 2m, an even number, where m is an integer.

When there are k+1 squares, there are four possible cases, since one square has four edges.

When the k+1th square shares only 1 edge with those old squares

the the resulting shape is 2m + 2, an even number.

When the k+1th square shares 2 edges with those old squares

the the resulting shape is 2m, an even number.

When the k+1th square shares 3 edges with those old squares

the the resulting shape is 2m-2, an even number.

When the k+1th square shares 2 edges with those old squares

the the resulting shape is 2m-4, an even number.

So, when there are n squares, no matter what n is, the resulting shape will always be an even number.

So, the length of the periphery is always even.

Proof for Question No.2:

Base case: when there is only one square, the length of the periphery of the resulting shape was 4, an even number.

Suppose when there are k squares, the length of the periphery of the resulting shape is 2m, an even number, where m is an integer.

When there are k+1 squares, there are four possible cases, since one square has four edges.

When the k+1th square shares only 1 edge with those old squares

the the resulting shape is 2m + 2, an even number.

When the k+1th square shares 2 edges with those old squares

the the resulting shape is 2m, an even number.

When the k+1th square shares 3 edges with those old squares

the the resulting shape is 2m-2, an even number.

When the k+1th square shares 2 edges with those old squares

the the resulting shape is 2m-4, an even number.

So, when there are n squares, no matter what n is, the resulting shape will always be an even number.

So, the length of the periphery is always even.