FIRST TUTORIAL SHEET – GABRIEL MOISE

Question 1

a) In the case where n<0 mult m n will be equal to m + mult m (n-1), as n/=0, so we will have to calculate mult m (n-1), where of course (n-1)<0, too. Recursively, at step k, k positive integer number, we will need to calculate mult m (n-k), where (n-k) is also less than 0, thus we will never get to a result for any k. So, if n<0 the program will need an infinite number of steps to complete the task.

b) We need to prove that P(n): mult m n = m*n is True for any given m and n>=0. We have the initial supposition:
P(0) : mult m 0 = 0 , where 0 is also equal to m*0 and it's True.
We will then suppose that P(n) is True and, based on that, we will prove that P(n+1) is also True, for any given m and n>=0.
So, from P(n) = True, we know that mult m n = m*n.
P(n+1): mult m (n+1) = m + mult m n (we supposed n>=0, so (n+1) > 0)
Using P(n), mult m (n+1) = m + m*n = m*(n+1), which means P(n+1) is also True.
So, by induction on n, for any given m and n>=0, mult m n = m*n.

c) For any given m, n>=0, mult m n needs (n+1) operations (n additions and one verification), as the function needs n steps to reach the mult m 0 form and one extra operation (verification) to know that n became 0. We can reduce the number of operations if, after each step, n becomes smaller in a quicker manner. For example:

```
newMult m n
        | n == 0 = 0
        | (n `mod` 2) == 0 = 2 * newMult m (n `div` 2)
        | otherwise = m + 2 * newMult m ((n-1) `div` 2)
```

This way, after each step, n becomes n/2 or (n-1)/2, which means we need [$\log_2 n$]+1 steps (the last one being the verification), which is definitely more time efficient than the first approach.

Question 2

a) We have the supposition P(n) = iter m n r = m*n+r, for any given m and r. We start the strong induction from P(0): iter m 0 r = m*0+r = r. That happens because if n==0, iter m n r returns r. So, P(0) is True, for any m and r. Now we suppose that for every 0<=i<=n we have P(i) = True, and we will prove that P(n+1) is also True. We distinguish two cases:

- If n is odd, let's say n=2*k+1, then n+1=2*(k+1), which is even, so

iter m (2*(k+1)) r = iter (m*2) (k+1) r
From the strong induction, we know that P(k+1) is True, because 0<=k+1<=n, as n=2*k+1, and instead of m, we have 2*m, which doesn't affect the supposition.
So, iter m (2*(k+1)) r = (m*2)*(k+1) + r = m*(2*(k+1))+r, so, in this case, P(n+1) is True.

- If n is even, let's say n=2*k, then n+1=2*k+1, which is neither 0, nor even, so

iter m (2*k+1) r = iter m (2*k) (r+m)

From the strong induction, we know that P(2*k) is True, because 0<=2*k<=n, as n=2*k, instead of m, we have 2*m and instead of r we have (r+m), which doesn't affect the supposition.
So, iter m (2*k+1) r = m*(2*k) + (r+m) = m*(2*k+1) + r, so, in this case, P(n+1) is True.
In conclusion, we showed that P(n+1) is True for any given m and r, so, by strong induction, iter m n r = m*n+r.

b) As fastMult m n = iter m n 0, we can conclude from a) that fastMult m n = m*n+0 = m*n
c) After each step, n is reduced to (n-1) or n/2 depending if n is odd or even, so in the worst case scenario, when n has the form 2^k-1, with k positive integer, we need 2*k steps to get to the n==0 case, as we need to reduce it by 1 and then reduce it by half each time, and n will go through these forms after each step : 2^k-1 -> 2^k-2 -> $2^{(k-1)}$-1 -> $2^{(k-1)}$-2 -> .. -> 2^2-1 -> 2^2-2 -> 2^1-1 -> 2^1-2 = 0. To reduce k to 0 we need 2*k and for the verification we need an extra step, so we will need 2*k+1 steps in total. As k is $\log_2$(n+1) in this worst case scenario, the maximum number of iterations of fastMult m n is [$\log_2$(n+1)]+1. I think that the program in efficient especially for big values of n, but it can be improved if instead of having two cases regarding the parity of n, we have 3 cases regarding the result of n `mod` 3. It can also be improved if we have 4, 5 or more cases, but this is a slightly significant improvement only for very big values of n.

Question 3

a) We will number the 6 chairs from 1 to 6. For the first chair, we have 6 possibilities, either one of the 6 people who attended the dinner party. For each of these possibilities, we furthermore have another 5 possibilities for the second chair, as the person who sits on the first seat of each case cannot also sit on the second one. Reasoning this way until we get to the 6th chair, we will get a total number of 6!=6*5*4*3*2*1 arrangements, which is equal to 720 different arrangements. In general, if we have n people and n chairs, we have n! different configurations, using the product rule.
b) For the first seat, we can have a man or a woman, and from there we can reason further ahead. So, if we have a man on the first seat, we will have 3 possibilities (there are 3 men). For each of these possibilities we need a woman on the second seat, so again 3 possibilities. For the third seat we need another male, so we are left with 2 possibilities, and for the fourth we need another woman, so again 2 possibilities. For the fifth and sixth chairs there are only one man and one woman remaining, so, by using the product rule we have 3*3*2*2*1*1=36 arrangements. But because we had a man on the first seat and there is an equal number of different ways of arranging the people around the table if a woman sits on the first chair, by following the same reasoning, the final answer is 36*2=72 ways. In general, if we have n couples, and we want to arrange them so that genders alternate around the table, we have 2*n*n*(n-1)*(n-1)*...*3*3*2*2*1*1 = 2*$(n!)^2$ different ways to do that.
c) For simplicity we will say that the couples are (A & a) (B & b) (C & c). So, for the first seat we can have either of the 6 people, but because we need to alternate the genders, and each person has 2 neighbors of the opposite gender who cannot be his/her partner, it means that the partner of each person is opposite to their seat. For example, if A sits on the second chair, then a sits on the 5th chair. Supposing we start on the first chair with a man, then we have 3 possibilities for that and for the fourth chair we need the woman who forms a couple with that man. For the second seat we have 2 women candidates as the other woman sits on the fourth chair, and on the fifth

chair will sit the men who forms a couple with the woman on the second chair. For the third chair we can only have the man who wasn't assigned a chair yet, and for the sixth we have the woman who forms a couple with him, who wasn't assigned a chair either. So here we have 3*2=6 possible configurations. As we started with a man, we have to multiply the result by 2 because the cases where we start with a random woman on the first seat are exactly the same. So, we have in total we have 6*2=12 viable configurations. *Question: How can we calculate the number of arrangements for any number n of couples? How can we obtain a general formula? I calculated by brute-force the result for 4 couples, which is 96, and for 5 couples, which is 3120, but I struggle to find the formula for the general case, and from 6 it becomes impossible (too many operations) to find a result that could guide me.*

Question 4

a) After tossing the fair coin three times, we obtain a configuration of 3 outcomes, their options being Heads or Tails, so we have a total of 2^3=8 number of possible configurations by using the product rule (HHH, HHT, HTH, HTT, THH, THT, TTH, TTT). The probability of obtaining the HTH configuration is 1/8 because we only have one favorable case out of 8 possible cases. In general, if we want to calculate the probability of obtaining a specific configuration after tossing the fair coin n times, it will be $1/2^n$, as there will be a number of 2^n total cases and only one favorable one.

b) As the configuration contains an odd number of tosses, let's say n = 2*k+1, and we obtain x Heads and y Tails, so x + y = 2*k+1, then there will always be more Heads or more Tails for a configuration, as x cannot be equal to y (obviously if x=y then we will have 2*x=2*k+1, with x and k positive integers, which is wrong). For each case where there are more Heads than Tails, we can get a symmetrical configuration by replacing every Heads with Tails and vice-versa. This way we split the configurations into 2 groups, one where there are more Heads than Tails and one where there are more Tails than Heads. For each configuration from group 1 we have a unique symmetrical configuration in group 2 and all configurations have a symmetric. So, there are 2^3/2=4 configurations in each group, so the probability is 4/8=1/2. In general, the probability of obtaining more Heads than Tails is still 1/2 for odd n (n = the number of tosses) because we get to form the 2 corresponding groups one with more Heads than Tails, and one with less. Howerever, for even n = 2*k, the number of favorable cases is equal to the sum of combinations of n by 0, 1, 2 … (k-1), for each case (for 0 Tails, for 1 Tails … for (k-1) Tails). So the number of favorable cases is ($2^{(2*k)}$ – (2*k)! / $(k!)^2$)/2 and the probability is the number of favorable cases over 2^n.

c) By tossing the fair coin 99 (odd number) times, we still have a 1/2 probability of obtaining more Heads than Tails, as we can split the configurations in 2 groups consisting of 2^{98} configurations, each of them having a symmetric correspondent in the other group, as stated at b).