Question 1

Proof:

(1) Let P(n) be the predicate "when the input to mySum() is n, this function terminates and returns 2n(n+1)(2n+1)." We will prove P(n) for all $n \geq 1$.

(2) Base Case: Assume the input to mySum() is 1. Then the condition on line 3 evaluates to true, so line 4 is executed and the function returns 12. Note that,

$$2n(n+1)(2n+1) = 2(1)(1+1)(2(1)+1) = 2(2)(3) = 12$$

So, P(1) is true.

(3) Induction Step: Let $n \geq 1$ be arbitrary and assume P(n). We will prove P(n+1), which states

$$2(n+1)(n+1+1)(2(n+1)+1)$$

(4) Suppose mySum() is called with input n+1. Then the parameter k has value $n+1 > 1$ when line 3 is reached, so the if-condition on line 3 evaluates to false. Thus line 4 is skipped and line 6 will be executed, so the parameter k still has value n+1 when line 6 is reached.

(5) In line 6, mySum() is called with input n+1-1 = n. By the inductive hypothesis, this function terminates and returns the value 2n(n+1)(2n+1). Thus, the value returned on line 6 is

$$2n(n+1)(2n+1) + 12(n+1)(n+1)$$

$$= (n+1)(2n(2n+1) + 12(n+1))$$

$$= (n+1)(4n^2 + 2n + 12n + 12)$$

$$= (n+1)(4n^2 + 8n + 6n + 12)$$

$$= (n+1)(4n(n+2) + 6(n+2))$$

$$= (n+1)(n+2)(4n+6)$$

$$= 2(n+1)(n+2)(2n+3)$$

$$= 2(n+1)(n+1+1)(2(n+1)+1)$$

Proving P(n+1).

(6) So, by induction, P(n) is true for all $n \geq 1$. Hence the given function is fully correct.

Question 2

(a)

LoopInv: $(j \leq N+1) \wedge (S = 2j(j-1)(2(j-1)+1))$

Proof: That LoopInv is a loop invariant

(1) Let $N \geq 1$ be arbitrary. Assume that pre is true in line 1. Let P(n) be the predicate "if the loop condition is being checked for the n-th time, then LoopInv is true." We will prove $\forall n \geq 1$, P(n).

(2) Base Case: $n = 1$. After line 2 executes j has value 1, and after line 3 executes S has value 0. Thus, $j_1 = 1$ and $S_1 = 0$ when line 5 is reached for the first time.

(3) By the precondition $N \geq 1$, which implies $j_1 \leq N+1$. Further, $S_1 = 0$ and $2j_n(j_n-1)(2(j_n-1)+1) = 2(1)(0)(2(0)+1) = 0$. So,

$$S_1 = 2j_1(j_1-1)(2(j_1-1)+1)$$

This proves that LoopInv is true when line 5 is reached for the first time, i.e: P(1) is true.

(4) Induction Step: Let $n > 1$ be arbitrary and assume P(n). That is, assume $j_n \leq N+1$ and $S_n = 2j_n(j_n-1)(2(j_n-1)+1)$. We will prove P(n+1), that is,

$$j_{n+1} \leq N+1 \wedge S_{n+1} = 2j_{n+1}(j_{n+1}-1)(2(j_{n+1}-1)+1)$$

(5) Suppose the loop condition is being checked for the n+1st time. Then it was previously checked for the n-th time and evaluated to true. Thus $j_n \leq N$. Further, by the inductive hypothesis, LoopInv was true when the loop condition was checked for the n-th time, so $S_n = 2j_n(j_n-1)(2(j_n-1)+1)$.

(6) Within the while-loop, the only line that changes the value of j is line 7, which increments it by 1. So, $j_{n+1} = j_n + 1$. From (5), $j_n \leq N$, so, $j_{n+1} = j_n + 1 \leq N+1$.

(7) Within the while-loop, line 6 is the only line that changes the value of S, which increases it by $3(2j)(2j) = 12j^2$. From (6), $j_n = j_{n+1} - 1$. So,

$$S_{n+1} = S_n + 12j_n^2$$

$$= 2j_n(j_n - 1)(2(j_n - 1) + 1) + 12j_n^2 \quad byI.H$$

$$= 2(j_{n+1} - 1)((j_{n+1} - 1) - 1)(2((j_{n+1} - 1) - 1) + 1) + 12(j_{n+1} - 1)^2$$

$$= 2(j_{n+1} - 1)(j_{n+1} - 2)(2j_{n+1} - 3) + 12(j_{n+1}^2 - 2j_{n+1} + 1)$$

$$= (2j_{n+1} - 2)(2j_{n+1}^2 - 4j_{n+1} - 3j_{n+1} + 6) + 12(j_{n+1}^2 - 2j_{n+1} + 1)$$

$$= 4j_{n+1}^3 - 4j_{n+1}^2 - 14j_{n+1}^2 + 14j_{n+1} + 12j_{n+1} - 12 + 12j_{n+1}^2 - 24j_{n+1} + 12$$

$$= 4j_{n+1}^3 - 6j_{n+1}^2 + 2j_{n+1}$$

$$= 2j_{n+1}(2j_{n+1}^2 - 3j_{n+1} + 1)$$

$$= 2j_{n+1}(2j_{n+1}^2 - 2j_{n+1} - 1j_{n+1} + 1)$$

$$= 2j_{n+1}(2j_{n+1}(j_{n+1} - 1) - 1(j_{n+1} - 1))$$

$$= 2j_{n+1}(2j_{n+1} - 1)(j_{n+1} - 1)$$

$$= 2j_{n+1}(j_{n+1} - 1)(2(j_{n+1} - 1) + 1)$$

as needed. Thus, P(n) $\implies$ P(n+1).

(8) By (6) and (7), P(1) $\wedge$ (P(n) $\implies$ P(n+1)), so, by induction, LoopInv is a loop invariant, specifically stating that

$$(j \leq N + 1) \wedge (S = 2j(j - 1)(2(j - 1) + 1))$$

(b)

Proof: That the program is partially correct

(1) Assume that LoopInv is a loop invariant and that line 8 is reached. We will show that post is true when line 8 is reached.

(2) Since line 8 was reached, the loop condition was checked for a final time and it evaluated to false. So, $j > N \implies j \geq N + 1$ when line 8 is reached. Since LoopInv is a loop invariant, it was true when this final check occurred, so $(j \leq N + 1) \wedge (S = 2j(j-1)(2(j-1)+1))$ when line 8 is reached.

(3) Since $j \geq N + 1$ and $j \leq N + 1$, it must be true that $j = N + 1$. Now,

$$S = 2j(j-1)(2(j-1)+1)$$

$$= 2(N+1)(N+1-1)(2(N+1-1)+1)$$

$$= 2N(N+1)(2N+1)$$

$$= (2N^2 + 2N)(2N+1)$$

$$= 4N^3 + 4N^2 + 2N^2 + 2N$$

$$= 4N^3 + 6N^2 + 2N$$

proving that the post condition is true. Thus, the program is partially correct.

Question 3

$E = 6a - b + 25$

Proof: That E is a loop measure

(1) Let $n \geq 1$ be arbitrary. We claim that $E_{n+1} \leq E_n - 1$.

(2) If the loop condition is being checked for the n+1st time, then it was previously checked for the n-th time and evaluated to true. When line 3 is reached for the n-th time, either $a_n > 0$ or $a_n \leq 0$. Proceed by cases.

(3) Case 1: $a_n > 0$. Then the condition on line 3 evaluates to true, so lines 4 and 5 are executed and line 7 is skipped. In line 4, a_{n+1} is assigned the value $a_n - 4$. In line 5, b_{n+1} is assigned the value $b_n - 20$. So,

$$E_{n+1} = 6a_{n+1} - b_{n+1} + 25$$

$$= 6a_n - 24 - b_n + 20 + 25$$

$$= 6a_n - b_n + 25 - 4$$

$$= E_n - 4 < E_n - 1$$

(4) Case 2: $a_n \leq 0$. Then the condition on line 3 evaluates to false, so lines 4 and 5 are skipped and line 7 is executed. In line 7, b_{n+1} is assigned the value $b_n + 3$ and a_{n+1} is not modified, so $a_{n+1} = a_n$. Now,

$$E_{n+1} = 6a_{n+1} - b_{n+1} + 25$$

$$= 6a_n - b_n - 3 + 25$$

$$= E_n - 3 < E_n - 1$$

(5) By (3) and (4), in all cases, $E_{n+1} \leq E_n - 1$.

(6) Now, suppose that $E \leq 0$. We claim that the loop condition evauluates to false. That is, $a \leq 0 \wedge b \geq 5$.

(7) By the given loop invariant, $a \geq -3$, and $b \leq 7 \implies -b \geq -7$, which together imply that

$$E = 6a - b + 25 \geq 6(-3) - 7 + 25 = 0$$

By supposition, $E \leq 0$, so $E = 0$. However, this can only occur if $a = -3$ and $b = 7$, so $a \leq 0$ and $b \geq 5$, as needed. Therefore, when $E \leq 0$, the loop evaluates to false.

(8) Thus, by (5) and (7), $E = 6a - b + 25$ is a loop measure.

Question 4

```
1. //Pre: n >= 1
2. x <-- 2*n
3. sum <-- 0
4. //LoopInv: x is even
5. while(x > 0)
6.      y <-- 0
7.      while(y < x)
8.          sum <-- sum + 3*y
9.          y <-- y + 1
10.     x <-- x - 2
```

(a) Find $f(n)$, the number of steps that this program executes as a function of n.

For the outer loop, x starts at $2n$ and decreases by 2 until it it executes its final iteration when $x = 2$, since x is even from the loop invariant. So, $x = 2n, 2n-2, 2n-4, ..., 4, 2$. Let $k = x/2$, then k starts at n and decreases by 1 until $k = 1$. Since addition is commutative, the sum from n decreasing until 1 is equal to the sum from 1 increasing until n.

For the inner loop, y starts at 0 and increases by 1 until it it executes its final iteration when $y = x - 1 = 2k - 1$ since $x = 2k$. So, $y = 1, 2, 3, ..., 2k-2, 2k-1$. So, the number of steps exectuted by the inner loop is

$$\sum_{y=0}^{2k-1}(3) + 1 = 6k + 1$$

, since lines 7, 8, and 9 are executed each iteration, and then we add 1 for the final time the loop condition is checked and evaluates to false.

Then, the number of steps executed by the entire program is

$$\sum_{k=1}^{n}(3 + 6k + 1) + 1 + 2$$

Since lines 5, 6, and 10 are executed once per outer loop iteration, the inner loop is executed once per outer loop iteration, and then add the final check of the loop condition when it evaluates to false as well as when lines 2 and 3 are executed at the beginning of the program. We can further simplify this to

$$= 6\sum_{k=1}^{n}(k) + \sum_{k=1}^{n}(4) + 3$$

$$= \frac{6n(n+1)}{2} + 4n + 3$$

$$= \frac{6n^2 + 6n + 8n + 6}{2}$$

$$= 3n^2 + 7n + 3$$

Thus, a function $f(n)$ that calculates the number of steps that this program executes as a function of n is

$$f(n) = 3n^2 + 7n + 3$$

(b) Calculate the value of the variable *sum* when the program terminates as a function of the input n.

We can utilise our previous formulas and modify them to count the value of the variable *sum* instead. To start, the inner loop is the only time the value sum is modiifed, where $sum = sum + 3y$. So, we can then adjust our inner loop in the formula and keep the values for the summations the same. This gives us

$$\sum_{k=1}^{n}(\sum_{y=0}^{2k-1}(3y))$$

$$= 3\sum_{k=1}^{n}(\sum_{y=0}^{2k-1}(y)$$

$$= 3\sum_{k=1}^{n}(\frac{(2k-1)(2k)}{2})$$

$$= 3\sum_{k=1}^{n}(2k^2 - k)$$

$$= 3[\frac{2n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}]$$

$$= \frac{2n(n+1)(2n+1) - 3n(n+1)}{2}$$

$$= \frac{2n(n+1)(2n+1) - 3n(n+1)}{2}$$

$$= \frac{(2n^2+2n)(2n+1)-3n^2-3n}{2}$$

$$= \frac{4n^3+4n^2+2n^2+2n-3n^2-3n}{2}$$

$$= \frac{4n^3+3n^2-n}{2}$$

Thus, a function $g(n)$ that calculates the value of the variable *sum* as a function of the input n is

$$g(n) = \frac{4n^3+3n^2-n}{2}$$

Question 5

Find a function $f(n)$ that counts the number of times that $fnc()$ is called as a function with the input n.

(1) Assume $n \geq 3$ and $\exists a \in \mathbb{N}, n = 3^a$.

(2) From the loop invariant, we know that $x = 3^r$.

(3) The outer loop starts at $x = 3$ and is then multiplied by 3 each iteration until it is executed for the last time when $x = n$, since $\exists a \in \mathbb{N}, n = 3^a$ from (1). So, $x = 3, 9, 27, 81, ..., 3^a$. Similarly, r starts at $r = 1$ and increases by one each iteration, and by (2), $x = 3^r \implies r = \log_3 x$, so we get $x = 3^1, 3^2, 3^3, ..., 3^a$, and thus $r = 1, 2, 3, ..., a$. But, from (1), $n = 3^a \implies a = \log_3(n)$. So the number of iterations from the outer loop is

$$\sum_{r=1}^{log_3 n} (1)$$

(4) Now, from (2), since $x = 3^r$, clearly x must be odd.

(5) Next, the inner loop starts at $j = 0$ and increases by 2 until it executes for the last time when $j = x - 1$, since j is even and from (4), x is odd. So, $j = 0, 2, 4, .., x - 1$. But, from (2), we get $j = 0, 2, 4, ..., 3^r - 1$. Now, let $k = j/2$, then $k = 0, 1, 2, 3, ..., (3^r - 1)/2$. Thus, the number of iterations of the inner loop is

$$\sum_{k=0}^{\frac{3^r-1}{2}} (1) = \frac{3^r-1}{2} - 0 + 1 = \frac{3^r+1}{2}$$

(6) Finally, since the loops are nested, and $fnc()$ is called once every inner loop iteration, the number of times $fnc()$ is called is

$$\sum_{r=1}^{log_3 n} (\frac{3^r + 1}{2})$$

$$= \frac{1}{2}[\sum_{r=1}^{log_3 n} (3^r) + \sum_{r=1}^{log_3 n} (1)]$$

$$= \frac{1}{2}[\frac{3^{\log_3(n)+1} - 1}{2} - 3^0 + \log_3(n)]$$

$$= \frac{1}{2}[\frac{(3^{\log_3(n)})(3^1) - 1}{2} - 1 + \log_3(n)]$$

$$= \frac{1}{2}[\frac{3n - 1}{2} - 1 + \log_3(n)]$$

$$= \frac{1}{2}[\frac{3n - 3}{2} + \log_3(n)]$$

(7) Thus, a fucntion $f(n)$ that counts the number of times that $fnc()$ is called as a funciton with the input n is

$$f(n) = \frac{1}{2}[\frac{3n - 3}{2} + \log_3(n)]$$