Q1. On pages 55 and 56 of the textbook there is a proof for the correctness of the program `avg`. The author used the invariant

$$Inv(\texttt{i}, \texttt{sum})\colon 0 \leq \texttt{i} < len(A) \wedge \texttt{sum} = \sum_{k=0}^{\texttt{i}-1} A[k].$$

As you can see, the invariant is a predicate in two variables: `i`, `sum`. These two variables are used in the program `avg`, but neither is the variable on which the induction proof is based. The author is using simple induction, but it is not very clear what the induction variable is (also the induction predicate itself is ambiguous). We want to make sure you understand what is going on there by having you re-write the proof by yourself in a style similar to the one we use in lectures. Here is the predicate you should be proving:

$$Q(j) : \text{At the beginning of the } j^{th} \text{ iteration}, \texttt{sum} = \sum_{k=0}^{\texttt{i}-1} A[k].$$

Remark 1: By the program's design, the variables `i`, `sum` may change with each iteration (in other words, both are functions of j). This is why it might be more appropriate to write

$$Q(j) : \texttt{sum}_j = \sum_{k=0}^{\texttt{i}_j-1} A[k] \quad \text{or} \quad Q(j) : \texttt{sum}(j) = \sum_{k=0}^{\texttt{i}(j)-1} A[k].$$

where $\texttt{i}_j$ (or $\texttt{i}(j)$) means the value of program variable `i` at the beginning of the j^{th} iteration (the same with `sum`). That said, we believe that the version above Remark 1 is the best to work with as long as one understands that `i`, `sum` are iteration dependent.

Remark 2: Using j as an index in Remark 1 has a different meaning from that which is intended by the author. The author is using indices to differentiate between the values of `i`, `sum` at the beginning of an (arbitrary) iteration and their values after the iteration is run. The end goal is to prove $Q(len(A))$. A proof by induction will show that

$$\forall j \in \{1, \ldots, len(A)\}, Q(j).$$

Your proof must follow the style used in lectures.

Proof. It will be shown using simple induction that $\forall j \in \{1, ..., \texttt{len}(A)\}, Q(j)$.

Base Case. Let $j = 1$. At the beginning of the first iteration of the loop, $i = 0$ and $sum = 0$. Notice that in the expression $\sum_{k=0}^{-1} A[k]$, the lower bound is greater than the upper bound. This results in an empty sum that evaluates to 0. Therefore the base case holds.

Induction Hypothesis. Suppose that for some $j \in \{1, ..., \texttt{len}(A)\}$, $Q(j)$ is true.

Induction Step. i increases by 1 at the end of each iteration, so at the beginning of the jth iteration, $i = j - 1 < \texttt{len}(A)$, which means that the loop body is entered. From the induction hypothesis,

$$sum = \sum_{k=0}^{j-2} A[k]$$

Running through the loop, on line 9, $A[j-1]$ is added to *sum*. After this line is executed, the new value of sum becomes

$$sum = \sum_{k=0}^{j-2} A[k] + A[j-1] = \sum_{k=0}^{j-1} A[k]$$

Then, line 10 executes and the value of i is incremented and becomes j. After the line is executed, the program returns to the top of loop, signifying the beginning of the $j+1$th iteration. *sum* remains unchanged since the execution of line 9, and is equal to

$$\sum_{k=0}^{j-1} A[k] = \sum_{k=0}^{i-1} A[k]$$

as desired.

By the principle of simple induction, at the beginning of the jth iteration, $Q(j)$ holds for any $j \in \{1, 2, ..., \texttt{len}(A), \texttt{len}(A)+1\}$.

At the beginning of the $\texttt{len}(A)+1$th iteration, $i = \texttt{len}(A)$, so the loop terminates. The program returns $sum/\texttt{len}(A)$ on line 11, which is exactly the average of the numbers in A.

□

Q2. –Following up on the Q1– We mentioned this in class, but it is good to remind you again of the fact that, proving $\forall j \in \{1, \ldots, len(A)\}, Q(j)$ is equivalent to proving $\forall n \in \mathbb{N}, Q'(n)$ where

$$Q'(n) : 0 \leq n < len(A) \Rightarrow Q(n+1).$$

Explain the equivalence.

Proof. It will be proven that the two statements imply each other, that is,

$$\forall j \in \{1, ..., \texttt{len}(A)\}, Q(j) \iff \forall n \in \mathbb{N}, 0 \leq n < \texttt{len}(A) \implies Q(n+1)$$

To prove the forward direction, suppose that $Q(j)$ holds true for all $j \in \{1, ..., \texttt{len}(A)\}$. Let n be a natural number such that $0 \leq n < \texttt{len}(A)$. It follows that $n \in \{0, 1, ..., \texttt{len}(A) - 1\}$, which implies that $n + 1 \in \{1, ..., \texttt{len}(A)\}$. By the original assumption, $Q(n + 1)$ is true.

Conversely, suppose that for all natural n, if $0 \leq n < \texttt{len}(A)$ then $Q(n + 1)$ is true. Let $j \in \{1, ..., \texttt{len}(A)\}$. Then $j - 1 \in \{0, ..., \texttt{len}(A) - 1\}$. It is clear that $j - 1$ is a natural number and that $0 \leq j < \texttt{len}(A)$. By the assumption in the beginning, $Q(j)$ is true.

Thus both statements are equivalent to each other.

□

Q3. Solve questions 6 to 10 on pages 64 to 66 of the textbook. There is a typo in Q8 line 12 (it should be `c = 1` instead of `c == 1`). Proofs must follow the lecture's format and level of clarity. Only **one** of the five questions will be selected for marking. A serious attempt of all the mentioned questions is a necessary condition for receiving more than 0 points.

Question 6.

1. Give a loop invariant that characterizes the values of a and y.

   Proof. Define the loop invariant to be

   $$Inv(x, y) : a = x + \frac{y(y+1)}{2} - 55$$

   It will be proven that at the beginning of the ith iteration of the loop, $Inv(x, y)$ is true.

   Base Case. At the beginning of the first iteration, $a = x$ and $y = 10$. Thus

   $$x + \frac{y(y+1)}{2} - 55 = x - \frac{10(11)}{2} - 55 = x - 55 + 55 = x = a$$

   The base case has been proven to be true.

   Induction Hypothesis. Now suppose that for some positive natural j, $Inv(a, y)$ holds true at the beginning of the jth iteration. y is initially 10 and is decremmented at the end of every iteration, so at the beginning of the jth iteration, $y = 10 - (j-1) = 11 - j$. Then the loop invariant states that

   $$a = x + \frac{(11-j)(12-j)}{2} - 55$$

   Induction Step. If $a \leq 0$, the loop terminates and never reaches the beginning of the $j+1$th iteration. Otherwise, on line 6 in the jth iteration, a is subtracted by $y = 11 - j$. Using the induction hypothesis, the value of a is now

   $$a = x + \frac{(11-j)(12-j)}{2} - 55 - (11-j) = x + \left(\frac{(11-j)(12-j)}{2} - (11-j)\right)$$

   $$= x + \frac{(11-j)(12-j-2)}{2} - 55 = x + \frac{(10-j)(11-j)}{2} - 55$$

   Finally, on line 7, the value of y is decremented by 1 and becomes $y = 10 - j$. The program then arrives at the beginning of the $j+1$th iteration. Indeed, it is true that

   $$a = x + \frac{(10-j)(11-j)}{2} - 55 = x + \frac{y(y+1)}{2} - 55$$

   Thus $Inv(a, y)$ is true at the beginning of the $j+1$th iteration. Verifying the validity of the loop invariant.

   □

2. Show that sometimes this code fails to terminate

Proof. Let $x = 56$. It wil be shown using simple induction that at the beginning of every iteration, $a > 0$, which implies that the loop can never terminate.

Consider the beginning of an arbitrary ith iteration. It is known that $y = 11 - j$. From the loop invariant found in part (a),

$$a = x + \frac{y(y+1)}{2} - 55 = 56 + \frac{(11-j)(12-j)}{2} - 55 = 1 + \frac{(11-j)(12-j)}{2}$$

Note that $\frac{(11-j)(12-j)}{2} < 0$ if and only if $11 < j$ and $j < 12$ at the same time, which is impossible for $j \in \mathbb{N}$. Thus $\frac{(11-j)(12-j)}{2} \geq 0$ so

$$a = 1 + \frac{(11-j)(12-j)}{2} \geq 1 > 0$$

Therefore, $a > 0$ at the beginning of all iterations, which means that the loop never terminates.

□

Question 7. State the pre- and post-conditions that the algorithms must satisfy, then prove that each algorithm is correct.

(a) *Proof.* Pre-condition: a is a non-zero number and b is a natural number.

Post-condition: Returns the value of a^b.

The correctness of the program will be proved using complete induction on b. For the entirety of the proof, a is fixed to be an arbitrary non-zero number.

Base Case. When $b = 0$, the method returns 1, which is exactly a^b.

Induction Hypothesis. Suppose that the program returns the correct value of a^b for all $b \leq n$, for some $n \in \mathbb{N}$.

Induction Step. When $b = n + 1$, $b \geq 1$, so line 3 is not reached. Consider two possible cases:

If b is even, line 5 and 6 will be executed. A new variable x is initialized to be equal to `exp_rec(a, b / 2)`. Since $\frac{b}{2}$ is natural and less than or equal to n, By the induction hypthesis, the value of x is $a^{\frac{b}{2}}$. The program returns `x * x`, which is equal to $a^{\frac{b}{2}} \cdot a^{\frac{b}{2}} = a^b$ on line 6.

If b is odd, line 8 and 9 will be executed. By a similar argument as the case above, a variable x is initialized to the value $a^{\frac{b-1}{2}}$ and `x * x * a` is returned, which has the value of $a^{\frac{b-1}{2}} \cdot a^{\frac{b-1}{2}} \cdot a = a^b$

Therefore, it can be concluded that the program satisfies the postconditions for all natural b.

□

(b) *Proof.* Let a be a non-zero number and b be a natural number. For a positive natural i, define the loop invariant

$$P(i) : \texttt{mult}^{\texttt{exp}} \cdot \texttt{ans} = a^b$$

It will be shown using induction on i that for $i \in \{1, 2, ..., \lfloor \log_2(b) \rfloor + 1\}$, $P(i)$ is true at the beginning of the ith iteration.

Base Case. Let $i = 1$. At the start of the very first iteration, `mult = a`, `ans = 1`, and `exp = b`, so it follows that

$$\texttt{mult}^{\texttt{exp}} \cdot \text{ans} = a^b \cdot 1 = a^b$$

Thus the base case holds.

Induction Hypothesis. Let $j \in \{1, 2, ..., \lfloor \log_2(b) \rfloor\}$. Suppose that at the beginning of the jth iteration, $P(j)$ is true. That is,

$$\texttt{mult}^{\texttt{exp}} \cdot \texttt{ans} = a^b$$

Induction Step. Let $\texttt{mult}_0, \texttt{exp}_0, \texttt{ans}_0$ be the value of $\texttt{mult}, \texttt{exp}, \texttt{ans}$ at the beginning of the jth iteration and let $\texttt{mult}_1, \texttt{exp}_1, \texttt{ans}_1$ denote the value of the variables at the beginning of the $j + 1$th iteration.

Now, two cases will be considered for the analysis of the loop execution.

If `exp` is odd, then line 7 will run and multiply `ans` by `mult`, which implies that $\texttt{ans}_1 = \texttt{ans}_0 \cdot \texttt{mult}_0$.

Line 8 will run and will multiply `mult` by itself, so $\texttt{mult}_1 = \texttt{mult}_0^2$.

As well, `exp` can be rewritten as $2n + 1$, for some natural n, so line 9 will result in $\texttt{exp}_1 = \texttt{exp}_0 // 2 = \frac{\texttt{exp}_0 - 1}{2}$

After line 9, the current iteration ends and the $j + 1$th iteration begins. Using algebriac manipulation and the induction hypothesis, $P(j + 1)$ can be verified to be true:

$$\texttt{mult}_1^{\texttt{exp}_1} \cdot \texttt{ans}_1 = (\texttt{mult}_0^2)^{\frac{\texttt{exp}_0 - 1}{2}} \cdot (\texttt{ans}_0 \cdot \texttt{mult}_0) = (\texttt{mult}_0)^{\texttt{exp}_0 - 1}(\texttt{mult}_0 \cdot \texttt{ans}_0)$$

$$= \texttt{mult}_0^{\texttt{exp}_0} \cdot \texttt{ans}_0 = a^b$$

Otherwise if `exp` is even, then line 7 will not run, so $\texttt{ans}_1 = \texttt{ans}_0$.

Finally, from line 8, it is seen that $\texttt{mult}_1 = \texttt{mult}_0^2$

`exp` can be written as $2k$ for some natural k, so line 9 produces $\texttt{exp}_1 = \texttt{exp}_0 // 2 = \frac{\texttt{exp}_0}{2}$.

The execution of line 9 denotes the end of the jth iteration and the start of the $j + 1$th iteraton. Similar to the previous case, it is seen that

$$\texttt{mult}_1^{\texttt{exp}_1} \cdot \texttt{ans}_1 = (\texttt{mult}_0^2)^{\frac{\texttt{exp}_0}{2}} \cdot (\texttt{ans}_0) = \texttt{mult}_0^{\texttt{exp}_0} \cdot \texttt{ans}_0 = a^b$$

Thus, in either case, $P(j + 1)$ has been shown to hold.

By the principle of simple induction, it has been shown that $P(\lfloor \log_2(b) \rfloor + 1)$ is true at the beginning of iteration number $\lfloor \log_2(b) \rfloor + 1$. Since `exp` was initialized as b and has been

halved $\lfloor \log_2(b) \rfloor + 1$ times, the value of `exp` at the beginning of the iteration is 0, which means that the loop terminates. Using the fact that $P(\lfloor \log_2(b) \rfloor + 1)$ holds,

$$\texttt{mult}^{\texttt{exp}} \cdot \texttt{ans} = \texttt{mult}^0 \cdot \texttt{ans} = \texttt{ans} = a^b$$

The program returns `ans` on line 10, which satisfies its postconditions. □

Question 8. Prove that `majority()` is correct.

Proof. Let A be a list with more than half of its entries equal to each other, and let x be that value. As well, let $n > \frac{\texttt{len}(A)}{2}$ represent the number of occurences of x in A. For a positive natural i, define the loop invariant

$$P(i) : \text{if } m \neq x, \text{ then } A[i : \texttt{len}(A)] \text{ has more than half of its entries equal to } x$$

It will be shown using complete induction that for $i \in \{1, 2, ..., \texttt{len}(A)\}$, $P(i)$ is true at the beginning of the ith loop iteration.

Base Case. Let $i = 1$. When the program first enters the loop, $m = A[0]$. Suppose that $m \neq x$. Then the length of $A[1 : \texttt{len}(A)]$ is reduced by 1, but still contains n occurences of x. This implies that $A[1 : \texttt{len}(A)]$ also has more than half of its elements equal to x. Thus the base case holds true.

Induction Hypothesis. Let $i = k$, for some $j \in \{1, 2, ..., \texttt{len}(A) - 1\}$. Suppose that for all $j \leq k$, at the beginning of the jth iteration, $P(j)$ is true. It is obvious that $k \leq \texttt{len}(A) - 1$, so the loop body runs.

Now, examine the contrapositive of the statement of the predicate:

if $A[j + 1 : \texttt{len}(A)]$ has less than or equal to half of its entries equal to x, then $m = x$

□

Question 9. Studying bubblesort.

(a) State and prove an invariant for the inner loop

(b) State and prove an invariant for the outer loop

(c) Prove that `bubblesort` is correct, according to its specifications

Question 10. Consider the following generalization of the `min` function.

(a) Prove that this algorithm is correct

(b) Analyze the worse-case running time of this algorithm.

Q4. Solve questions 6,7,10,12, and 14 on pages 46 to 48 of the textbook. Only **two** of the five questions will be selected for marking. A serious attempt of all the mentioned questions is a necessary condition for receiving more than 0 points.

Question 6. Let $T(n)$ be the number of binary strings of length n where every 1 is immediately preceded by a 0.

(a) Develop a recurrence for $T(n)$

(b) Find a closed form for $T(n)$

(c) Prove that the closed form is correct using induction

Question 7. Let $T(n)$ denote the number of distinct full binary trees with n nodes. Give a recurrence for $T(n)$. Then use induction to prove that $T(n) \geq \frac{1}{n}2^{(n-1)/2}$.

Question 10. Let $H(n)$ denote the number of binary strings of length n that have no odd length blocks of 1's.

Develop a recursive definition for $H(n)$, and justify why it is correct. Then find a closed form for H using repeated substitution.

Question 12. Analyze the worse-case runtime of `fast_rec_mult`.

Question 14. Recall the recurrence for the worst-case runtime of quicksort:

$$T(n) = \begin{cases} c, & \text{if } n \leq 1; \\ T(|L|) + T(|G|) + dn, & \text{if } n > 1; \end{cases}$$

where L and G are the partitions of the list.

1. Suppose the lists are always evenly split; that is, $|L| = |G| = \frac{n}{2}$ at each recursive call. Find a tight asymptotic bound on the runtime of quicksort using this assumption.

2. Now suppose that the lists are always very unevenly split: $|L| = n - 2$ and $|G| = 1$ at each recursive call. Find a tight asymptotic bound on the runtime of quicksort using this assumption.