1 Question 2

a)

It is clear that the algorithm runs in $O(|E|)$. We see that every time the while loop runs and picks an edge, it sets at least one of the two vertices at the endpoints of the edge to have a weight of 0 in constant time. Thus, this edge can no longer be picked again (this also means that the while loop can just pick an edge by iterating over a fixed list of edges). In the worst case, each edge is picked once, resulting in $|E|$ iterations of the while loop. Thus, the runtime is $O(c|E|) \in O(|E|)$.

```
# approximation for weighted vertex cover

# define Vertex and Edge data types (adjacency list representation)
    class Vertex(object):
        def __init__(self, neighbors, w):
            # A vertex contains a list of references to other
            # vertices (neighbors) and an weight (w).
            self.neighbors = neighbors
            self.w = w

    class Edge(object):
        def __init__(self, u, v):
            # An edge contains two references to its endpoint vertices.
            self.u = u
            self.v = v

    def weighted_vertex_cover(vertices, edges):
        # Finds an approx min weighted vertex cover
        # given a list of vertices and a list of edges (i.e. Graph).
        cover_size = 0
        for e in edges:
            # O(|E|)
            if e.u.w == 0 or e.v.w == 0: continue
            x = min(e.u.w, e.v.w)
            if e.u.w != e.v.w: cover_size += 1
            else cover_size += 2
            e.u.w -= x
            e.v.w -= x

        cover = [None] * cover_size
        i = 0
        for v in vertices:
            if v.w == 0:
                cover[i] = v
                i += 1

        return cover
```

b)

Consider the following graph: $V = \{u, v, w\}$ with weights $w(u) = 2, w(v) = 4, w(w) = 3$ and edges $E = \{(u, v), (v, w)\}$.

Then, without loss of generality, the algorithm in the question can first pick edge (u, v), setting $w(u) = 0$ and $w(v) = 2$. Then, edge (v, w) is picked since both of its endpoint vertices have positive weight. The edges picked by the algorithm are thus: $\{(u, v), (v, w)\}$. It is clear that the edges do not form a matching as v appears twice.

c)

Let $k = t - i$. Prove that $P(k) = w_{t-k}(A) \leq 2w_{t-k}(OPT)$ for $k = 0$ to $k = t$.

Base case: let's prove for $k = 0$ that $P(k = 0) = w_{t-k}(A) \leq 2w_{t-k}(OPT) = w_t(A) \leq 2w_t(OPT)$ is true. Observe that $w_t(A) = 0$ since at the end of the tth (last) iteration in the algorithm, we construct A by choosing all the vertices with 0 weight. Thus, naturally, the weight of A must be 0 in the tth iteration by A's construction. Note that all weights are always non-negative, since we only ever decrease the weight of a vertex by it's current value at the most. Thus, $2_w t(OPT) \geq 0$. Thus, we have $0 = w_t(A) \leq 2w_t(OPT)$.

Suppose that for some $k = n$ that $P(k = n) = w_{t-n}(A) \leq 2w_{t-n}(OPT)$ is true. This is our inductive hypothesis. Prove that $P(k = n + 1)$ is true. For the edge picked in the $t - n$th iteration, we know that at least one of the endpoint vertices must be in the cover A (the weight of at least one edge will be reduced to 0), or both (both vertices same weight). Thus, there are two situations: $w_{t-n}(A) = w_{t-n-1}(A) - x_{t-n}$ or $w_{t-n}(A) = w_{t-n-1}(A) - 2x_{t-n}$. Now, we know that since we pick a distinct edge in each iteration (by previous parts), then at least one of the vertices is in the optimal cover (by definition that a vertex cover must have one vertex from each edge). Thus, since are reducing the weight of both endpoint vertices in every iteration, and since at least one of those vertices are in the cover, we have two cases: $w_{t-n}(OPT) = w_{t-n-1}(OPT) - x_{t-n}$ or $w_{t-n}(OPT) = w_{t-n-1}(OPT) - 2x_{t-n}$.

Now, by the inductive hypothesis, $w_{t-n}(A) \leq 2w_{t-n}(OPT)$. Let's look at all

the different combination of cases:

$$
\begin{aligned}
&\text{Case 1 (one vertex in A, one vertex in OPT)} \\
&w_{t-n-1}(A) - x_{t-n} \leq 2(w_{t-n-1}(OPT) - x_{t-n}) \\
\implies &w_{t-n-1}(A) \leq 2w_{t-n-1}(OPT) - x_{t-n} \\
\implies &w_{t-n-1}(A) \leq 2w_{t-n-1}(OPT) \\
&\text{Case 2 (two vertices in A, one vertex in OPT)} \\
&w_{t-n-1}(A) - 2x_{t-n} \leq 2(w_{t-n-1}(OPT) - x_{t-n}) \\
\implies &w_{t-n-1}(A) \leq 2w_{t-n-1}(OPT) \\
&\text{Case 3 (one vertex in A, two vertex in OPT)} \\
&w_{t-n-1}(A) - x_{t-n} \leq 2(w_{t-n-1}(OPT) - 2x_{t-n}) \\
\implies &w_{t-n-1}(A) \leq 2w_{t-n-1}(OPT) - 3x_{t-n} \\
\implies &w_{t-n-1}(A) \leq 2w_{t-n-1}(OPT) \\
&\text{Case 4 (two vertices in A, two vertices in OPT)} \\
&w_{t-n-1}(A) - 2x_{t-n} \leq 2(w_{t-n-1}(OPT) - 2x_{t-n}) \\
\implies &w_{t-n-1}(A) \leq 2w_{t-n-1}(OPT) - 2x_{t-n} \\
\implies &w_{t-n-1}(A) \leq 2w_{t-n-1}(OPT)
\end{aligned}
$$

Thus, we see that in all cases, we get that $P(k = n + 1) = w_{t-(n+1)}(A) \leq 2w_{t-(n+1)}(OPT)$ is true.

Thus, by our base case and our inductive hypothesis, we know that $P(k) = w_{t-k} \leq 2w_{t-k}(OPT)$ for all integer $k \in [0, \ldots t]$. Since $k = t - i$, we can clearly rearrange to see that $w_i(A) \leq 2w_i(OPT)$ is true for all integer $i \in [0, t]$. And so, $w_0(A) \leq 2w_0(OPT)$ is true.