1. [10 points] Consider the following implementation of the palindrome checking problem (question 4 on assignment 1):

```cpp
bool isPalindrome(const string& s) {
    bool result = true;
    int size = s.size();
    int i = 0;

    while (result && i < size/2) {
        result = (s[i] == s[size-1-i]);
        i++;
    }
    return result;
}
```

a) [5 points] Give an expression for the worst case time (note not complexity) of this algorithm.

b) [5 points] Assuming that the strings contain only of the letters “a” and/or “b”, and that all strings are equally likely, what is the average case time for this algorithm?
2. [10 points] Consider the following algorithm that satisfies the specification as follows:

**Pre:** \(1 \leq m \leq n\)

**Post:** \(\text{result} \leq n - m \rightarrow (\forall i, 0 \leq i < m \rightarrow P[i] = S[\text{result} + i]) \land \text{result} > n - m \rightarrow (\exists j, 0 \leq j \leq n - m \land (\forall i, 0 \leq i < m \rightarrow P[i] = S[j + i]))\)

```
result = -1
matched = false
while (result < n - m \land \neg matched) do
    result = result + 1
    i = 0
    matched = true
    while (i < m \land matched) do
        matched = matched \land (P[i] == S[\text{result} + i])
        i = i + 1
    end while
end while
if (\neg matched) then
    result = result + 1
end if
```

a) [9 points] Give an expression for the exact worst case (i.e., \(\Theta\)) complexity for this algorithm. Show your workings.

b) [1 point] What does this tell us about the complexity of the problem solved by this algorithm?

3. [15 points] Give a (deterministic) finite state automata on the input language \(\Sigma = \{0, 1\}\) accepting each of the following languages:

a) [5 points] The set of all strings ending in 00.

b) [5 points] The set of all strings with three consecutive 0’s.

c) [5 points] The set of all strings such that every block of five consecutive symbols contains at least two 0’s.

4. [15 points] Let \(G\) be an undirected graph consisting of a set of nodes, \(N\), and a set of edges \(E \subseteq N \times N\). A set of nodes \(N' \subseteq N\) is called a vertex cover for \(G\), if for every edge in \(E\), at least one of its end-points is in \(N'\). The vertex cover problem is, given a graph, \(G\), and a positive integer, \(K\), determine whether there is a vertex cover for \(G\) with at most \(K\) nodes.

Show that the vertex cover problem is NP-complete.