1. Consider the following function where \( n \) is the size of \texttt{array}.

```c
1: void rotate(int array[], int n)
2: {
3:     for (int i = 0; i < n-1; ++i) {
4:         int tmp = array[0];
5:         for (int j = 1; j < n; ++j)
6:             array[j-1] = array[j];
7:         array[n-1] = tmp;
8:     }
9: }
```

(a). In plain English state whether the function has a best and worst case, and if so, what they are.

(b). Give the detailed cost \( T(n) \) of the function counting only the total number of array accesses.

(c). Give the cost of \( T(n) \) using big-O notation.

(d). Use the definition of big-O to show that your answer in (c) is correct. (That is, find suitable values for constants \( k \) and \( n_0 \)).
2. Consider the following function where \( n \) is the size of array.

1:     // pre: array has at least one element
2:     bool contains_sum(const int array[], int n, int s)
3:     {
4:         for (int i = 0; i < n; ++i) {
5:             int sum = array[i];
6:             for (int j = i + 1; sum <= s and j < n; ++j) {
7:                 sum += array[j];
8:                 if (sum == s)
9:                     return true;
10:             }
11:         }
12:         return false;
13:     }

(a). In plain English state whether the function has a best and worst case, and if so, what they are.

(b). Give the detailed cost \( T(n) \) of the function counting only the total number of times line 8 is called. Each call to line 8 should count as one unit time.

(c). Give the cost of \( T(n) \) using big-O notation.

(d). Use the definition of big-O to show that your answer in (c) is correct. (That is, find suitable values for constants \( k \) and \( n_0 \).)