Task Description:

Task-1: Count the number of primitive operations executed below and determine the best & the worst cases: <u>(1 points)</u>

```
Algorithm: $arrayMin(A, n)$
$currentMin \leftarrow A[0]$
$i \leftarrow 1$
while $i \leq n - 1$ do
    if $currentMin \geq A[i]$ then
            $currentMin \leftarrow A[i]$
    $i \leftarrow i + 1$
return $currentMin$
```

Best Case: O(4) if I>n-1 and it doesn't go into the loop
Worst case: O(3n+3) if it goes into the loop

Task-2: Determine the Big-O notation for: <u>(3 points)</u>

a) $2 + n(2 + 3n)$

O(n)

b) $n + 2(n + 3n)n + \frac{n}{2}$

O(n^2)

c) $n^3 \log n + 2n + 1 + 3n^2 + n(\log n)^2$

O(n^3*logn)

Task-3: Determine the Complexity Of The Following Small Functions: (6 points)

a)
```
for (i = sum = 0; i < n; i++)
    sum += a[i];
```

O(n)

b)
```
for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
a[i][j] = i*j;
```

O(n^2)

c)
```
for (i = n; i >= 1; i--)
    for (j = i; j <= n; j++)  /* Note that the value of the inner loop variable (j)  */
        ...                    /* depends on the value of the outer loop variable (i) */
```

O(n^2)

```
d)  for (i = 1; i <= n; i++)
        for (j = i; j <= i; j++)      /* Note that the value of the inner loop variable (j)  */
            ...                         /* depends on the value of the outer loop variable (i) */
```

O(n)

```
e)  for (i = 0; i < n; i++)
        for (j = n; j > 1; j/=2)
          ...
```

O(nlogn)

```
f)  int factorial (int n)
    {
        if (n <= 1)
          return 1;
        else
          return n * factorial(n-1);
    }
```

O(n)