Task Description:

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

Algorithm: *a*rrayMin(A, n)	
$currentMin \leftarrow$ A[0]	2
$i \leftarrow 1$	1
while $i \leq n-1$ **do**	n
if $currentMin \geq$ A[i] **then**	2(n-1)
$currentMin \leftarrow$ A[i]	2(n-1)
$i \leftarrow i+1$	2(n-1)
return $currentMin$	1

best cases: 2+1+n+2(n-1)+2(n-1)+1=5n
worst cases: 2+1+n+2(n-1)+2(n-1)+2(n-1)+1=7n-2

Task-2: Determine the Big-O notation for: (3 points)

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

o(n^2)

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^3logn)

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)