```
_start:

cout << "output: "
    la a0, str
    li a7, 4
    ecall

partition(arr, n, k)
    la a0, arr
    lw a1, n
    lw a2, k
    jal partition

cout << partition()
    li a7, 1
    ecall

    j end

end:
    # Exit program
    li a7, 10
    ecall
```

Testcase1

```
Console
output: 60
Program exited with code: 0
```

Testcase2

```
Console
output: 17
Program exited with code: 0
```

```
partition:                                   Non-Leaf
stack
    addi sp, sp, -28
    sw a0, 24(sp) # int arr[]
    sw a1, 20(sp) # int n
    sw a2, 16(sp) # int k
    sw ra, 12(sp) # return address

int lo = getMax(arr, n);
    jal getMax
    mv t1, a0
    # store stack
    sw t1, 8(sp) # lo

int hi = getSum(arr, n);
    lw a0, 24(sp)
    jal getSum
    mv t2, a0
    # store stack
    sw t2, 4(sp) # hi

while (lo < hi)
whileLoop:                                   Loop
    # restore stack
    lw t1, 8(sp) # lo
    lw t2, 4(sp) # hi
    bge t1, t2, breakWhileLoop

    # mid = lo + (hi - lo) / 2
    sub t3, t2, t1
    srai t3, t3, 1
    add t3, t1, t3
    # store stack
    sw t1, 8(sp) # lo
    sw t2, 4(sp) # hi
    sw t3, 0(sp) # mid

    # restore stack
    lw a0, 24(sp) # int arr[]
    lw a1, 20(sp) # int n
    mv a2, t3 # int maxLen
    jal numberOfPainters
    mv t4, a0

    # restore stack
    lw a2, 16(sp) # int k
    lw t1, 8(sp) # lo
    lw t2, 4(sp) # hi
    lw t3, 0(sp) # mid
    # requiredPainters > k
    bgt t4, a2, lo_mid_1
    # requiredPainters <= k
    mv t2, t3 # hi = mid
    # store stack
    sw t2, 4(sp) # hi
    beq zero, zero, whileLoop # loop
lo_mid_1:
    addi t1, t3, 1 # lo = mid + 1
    # store stack
    sw t1, 8(sp) # lo
    beq zero, zero, whileLoop # loop

breakWhileLoop:
    mv a0, t1 # put return value
    # restore stack
    lw ra, 12(sp) # return address
    jr ra
```

```
getMax:                                      Leaf
int max = INT_MIN;
    li t1, -2147483648

for (int i = 0; i < n; i++)
    li t2, 0 # i = 0
forLoop1:                                    Loop
    bge t2, a1, breakForLoop1
    slli  t3, t2, 2
    add   t3, a0, t3
    lw    t3, 0(t3)
        if (arr[i] > max)
    # arr[i] <= max
    ble t3, t1, bottom1
    # arr[i] > max
    mv t1, t3 # max = t3 = arr[i]
bottom1:
    addi t2, t2, 1 # i += 1
    beq zero, zero, forLoop1 # loop
breakForLoop1:
    mv a0, t1 # put return value
    jr ra
```

```
getSum:                                      Leaf
int total = 0;
    li t1, 0

for (int i = 0; i < n; i++)
    li t2, 0 # i = 0
forLoop2:                                    Loop
    bge t2, a1, breakForLoop2
    slli  t3, t2, 2
    add   t3, a0, t3
    lw    t3, 0(t3)
    add   t1, t1, t3 # total += arr[i]
    addi t2, t2, 1 # i += 1
    beq zero, zero, forLoop2 # loop
breakForLoop2:
    mv a0, t1 # put return value
    jr ra
```

```
numberOfPainters:                            Leaf
int total = 0, numPainters = 1;
    li t1, 0 # total = 0
    li t2, 1 # numPainters = 1

for (int i = 0; i < n; i++)
    li t3, 0 # i = 0
forLoop3:                                    Loop
    bge t3, a1, breakForLoop3

    slli  t4, t3, 2
    add   t4, a0, t4
    lw    t4, 0(t4)
    add   t1, t1, t4

    # total <= maxLen (mid)
    ble t1, a2, bottom3
    # total > maxLen (mid)
    mv t1, t4 # total = t4 = arr[i]
    addi t2, t2, 1 # numPainters++

bottom3:
    addi t3, t3, 1 # i += 1
    beq zero, zero, forLoop3 # loop
breakForLoop3:
    mv a0, t2 # put return value
    jr ra
```