1.

```
.data
A: .float 832040 1346269 2178309 3524578 5702887 9227465 14930352 24157817 39088169 63245986
102334155 165580141 267914296 433494437 701408733 1134903170 1836311903 2971215073
i: .word 4 # start index in A
g: .word 20  # how many words to collect from A counting from i  #then change g to 12
k: .word 18 # how many words total in A
msgoverflow:    .asciiz  "overflow!"
.text
.globl main

main:
# load registers for array A
la      $s0, A          # load address of A
lw      $s1, i          # load word i into s1
lw      $s2, g          # load word g into s2
lw      $s3, k          # load word g into s3


# initialize address of A[i]
sll $s6, $s1, 2
add $s6, $s6, $s0

#initialize a register where you will compute the sum of the elements
add $s4, $zero, $zero
add $s2, $s2, $s1

# in the loop iteratively grab successive elements of A and add to sum
```

```
Loop:
# loop through A, check index bounds
        slt $t0, $s1, $s3 #$t0 <- 1 if (startA < sizeA), $t0 <- 0 if (startA >= sizeA)
        beq $t0, $zero, end
        slt $t0, $s1, $s2 #$t0 <- 1 if (startA < startA+g), $t0 <- 0 if (startA >= sizeA)
        beq $t0, $zero, end
# load A[i]
        lw $a0, 0($s6)

        addu $a1, $s4, $zero

        # now call the ifoverflow function with 2 arguments: the sum and the content of A[i]
        ## remember to move sum from its register to the first '$a' register
        #and to move the content of A[i] from the load register to the 2nd '$a' register
        #check the result register: if the result is 1 jump out
        jal ifoverflow
        bne $v0, $zero, end
        # add the loaded word content to the sum (USE addu for addition)
        addu $s4, $s4, $a0
        # update the index  (you need it for the entrance checks)
        addi $s1, $s1, 1
        # update the address of A[i]
        add $s6,$s6, 4

        # loop again
        j Loop

end:
```

```
        beq $v0, $zero, normend
        la $a0, msgoverflow
        ori $v0, $zero, 4
        syscall

normend:
        addu $a0, $s4, $zero
        ori $v0, $zero, 1
        syscall
#then exit
        ori $v0, $zero, 10          #exit
        syscall

ifoverflow:

        addu    $t0, $a0, $a1    # $t0 <= sum of $a0 and $a1
        nor     $t3, $a0, $zero   # $t3 = NOT $a0
        sltu    $t3, $t3, $a1       # $t3 = 1 if $t3 < $a1
                # Overflow if $a0 + $a1 > (2^32 - 1)  # $t3 = 1  overflow
        bne     $t3, $zero, Overflow

        add     $v0, $zero $zero
        jr $ra

        Overflow:
        addi    $v0, $zero 1
        jr $ra
```

2.

```
.data

A: .float 832040.0 1346269.0 2178309.0 3524578.0 5702887.0 9227465.0 14930352.0 24157817.0
39088169.0 63245986.0 102334155.0 165580141.0 267914296.0 433494437.0 701408733.0
1134903170.0 1836311903.0

i: .word 0 # start index in A

g: .word 20 # how many words to collect from A counting from i  (can be any integer number)

k: .word 17 # how many words total in A

zero: .float 0.0

.text

.globl main


main:


# load registers for array A

la $s0, A  # load address of A

lw $s1, i  # load word i into s1

lw $s2, g  # load word g into s2

lw $s3, k  # load word g into s3




# initialize address of A[i]

sll $s6, $s1, 2

add $s6, $s6, $s0


add $s2, $s2, $s1   #where is our final word index
```

```
# initialize a FP register where you will compute the sum of the elements
 lwc1 $f0, zero   #you can use l.s as well

# in the loop iteratively grab successive elements of A and add to the sum
Loop:
# loop through A, check index bounds
        slt $t0, $s1, $s3 #$t0 <- 1 if (startA < sizeA), $t0 <- 0 if (startA >= sizeA)
        beq $t0, $zero, end
        slt $t0, $s1, $s2 #$t0 <- 1 if (startA < startA+g), $t0 <- 0 if (startA >= sizeA)
        beq $t0, $zero, end
# load A[i]   (USE appropriate load instruction)
        lwc1 $f1, 0($s1)
# add the loaded word content to the sum (use appropriate instruction for addition)
        add.s $f0, $f0, $f1
        addi $s1, $s1, 1
# update the address of A[i]
        add $s6,$s6, 4
        j Loop

#print result: move the sum register into an appropriate register for that

end:
mov.s $f12, $f0
ori $v0, $zero, 2
syscall

ori $v0, $zero, 10
syscall
```

3.

a)

1332979686

1332979686/2**30 (tip: instead of dividing by 2 consecutively, you can guess a power of 2 that divides to the normalized form, and then adjust)

1. 241434073075652122249755859375

- Depending on whether you rounded **1.241434073075652122249755859375**
- **as 1.24143407307 or 1.24143407308 you got an error of 102 or 26**
- other rounding errors are also acceptable

Algorithm:

0.241434073075652122249755859375 * 2 = 0.48286814615130424449951171875 -> 0

0.48286814615130424449951171875 * 2 = 0.96573629230260848999023437 5 -> 0

0.96573629230260848999023437 5 * 2 = 1.93147258460521697998046875 -> 1

0.93147258460521697998046875 * 2 = 1.86294516921043395996093 75 -> 1

0.86294516921043395996093 75 * 2 = 1.72589033842086791992187 5 -> 1

0.72589033842086791992187 5 * 2 = 1.45178067684173583984375 -> 1

0.45178067684173583984375 * 2 = 0.90356135368347167968 75 -> 0

0.90356135368347167968 75 * 2 = 1.80712270736694335937 5 -> 1

0.80712270736694335937 5 * 2 = 1.61424541473388671875 -> 1

0.61424541473388671875 * 2 = 1.2284908294677734375 -> 1

0.2284908294677734375 * 2 = 0.456981658935546875 -> 0

0.456981658935546875 * 2 = 0.91396331787109375 -> 0

0.91396331787109375 * 2 = 1.8279266357421875 -> 1

0.8279266357421875 * 2 = 1.655853271484375 -> 1

0.655853271484375 * 2 = 1.31170654296875 -> 1

0.31170654296875 * 2 = 0.6234130859375 -> 0

0.6234130859375 * 2 = 1.246826171875 -> 1

0.246826171875 * 2 = 0.49365234375 -> 0

0.49365234375 * 2 = 0.9873046875 -> 0

0.9873046875 * 2 = 1.974609375 -> 1

0.974609375 * 2 = 1.94921875 -> 1

0.94921875 * 2 = 1.8984375 -> 1

0.8984375 * 2 = 1.796875 -> 1

Sign Exponent Significand

0 10011101 00111101110011101001111

1.241434073075652122497558593750 (sub 10) = 1.00111101110011101001111

Represented number:

(1)*(1+0*2^-1+0*2^-2 + 2^-3+2^-4 + 2^-5+2^-6 + 2^-8+2^-9 + 2^-10+2^-13 + 2^-14+2^-15 + 2^-17+2^-20 + 2^-21+2^-22 +2^-23)*2*30 = 1332979584

Error: 1332979686 – 1332979584 = 102

b)

-0.8544921875

Multiply by 2, 1 time to get -1.708984375

realExp = -1

Bias = 127

Exp = -1 + 127 = 126 = 1111110

Algorithm:

0.708984375 * 2 = 1.41796875 -> 1

0.41796875 * 2 = 0.8359375 -> 0

0.8359375 * 2 = 1.671875 -> 1

0.671875 * 2 = 1.34375 -> 1

0.34375 * 2 = 0.6875 -> 0

0.6875 * 2 = 1.375 -> 1

0.375 * 2 = 0.75 -> 0

0.75 * 2 = 1.5 -> 1

0.5 * 2 = 1 -> 1

Sign Exponent Significand

1 1111110 10110101100000000000000

-1.708984375 (sub 10) = 1.10110101100000000000000

Represented number:

(-1)*(1+1*2^-1+1*2^-2)*2^1 = −0.875

Error is zero.