Отчет Дз_4

Тесты

```
**test 1**
Enter the number of elements 1 to 10: 3
Enter element: 1
——————— Elements left: 2
Enter element: 2
——————— Elements left: 1
Enter element: 3
——————— Elements left: 0
Sum = 6
Even numbers counter = 1
Odd numbers counter = 2
-- program is finished running (0) --

Enter the number of elements 1 to 10: 3
Enter element: -1
——————— Elements left: 2
Enter element: -2
——————— Elements left: 1
Enter element: -3
——————— Elements left: 0
Sum = -6
Even numbers counter = 1
Odd numbers counter = 2
-- program is finished running (0) --


**test 2**
Enter the number of elements 1 to 10: 0
!!Intenger is out of array size bounds!!
-- program is finished running (0) --

Enter the number of elements 1 to 10: -1
!!Intenger is out of array size bounds!!
-- program is finished running (0) --

Enter the number of elements 1 to 10: 11
```

```
!!Intenger is out of array size bounds!!
-- program is finished running (0) --


**test 3**
Enter the number of elements 1 to 10: 4
Enter element: 1000000000
------ Elements left: 3
Enter element: 1000000000
------ Elements left: 2
Enter element: 1000000000
------ Elements left: 1
Enter element: 1000000000
------ Elements left: 0
!!Sum overflow!! Last sum = 2000000000
Counted elements: 2
Even numbers counter = 4
Odd numbers counter = 0
-- program is finished running (0) --

Enter the number of elements 1 to 10: 3
Enter element: -1000000000
------ Elements left: 2
Enter element: -1000000000
------ Elements left: 1
Enter element: -1000000000
------ Elements left: 0
!!Sum overflow!! Last sum = -2000000000
Counted elements: 2
Even numbers counter = 3
Odd numbers counter = 0
-- program is finished running (0) --


**zero test**
Enter the number of elements 1 to 10: 3
Enter element: 0
------ Elements left: 2
Enter element: 0
------ Elements left: 1
Enter element: 0
------ Elements left: 0
Sum = 0
```

```
Even numbers counter = 3
Odd numbers counter = 0
-- program is finished running (0) --

**test 4**
Enter the number of elements 1 to 10: 4
Enter element: 10
------- Elements left: 3
Enter element: -20
------- Elements left: 2
Enter element: 10
------- Elements left: 1
Enter element: -5
------- Elements left: 0
Sum = -5
Even numbers counter = 3
Odd numbers counter = 1
-- program is finished running (0) --
```

Код

```
.data
msg_start:      .asciz "Enter the number of elements 1 to 10: "
msg_elem_in:    .asciz "Enter element: "
msg_elem_left:  .asciz "------- Elements left: "
msg_new_line:   .asciz "\n"
msg_sum:        .asciz "Sum = "
msg_arroverflow:.asciz "!!Sum overflow!! Last sum = "
msg_curr_count: .asciz "Counted elements: "
msg_error:      .asciz "!!Intenger is out of array size bounds!!"
msg_even:       .asciz "Even numbers counter = "
msg_odd:        .asciz "Odd numbers counter = "

max_elements:   .word 10
array:          .space 40

.text
main:
# output hello msg
la      a0, msg_start
li      a7, 4           # output str
ecall

# input size of array
```

```
li      a7, 5           # input int
ecall
mv      t0, a0          # save size of arr in t0 from a0

#  bounds of array
li      t1, 1
li      t2, 10
# check ≥1
blt     t0, t1, error
# check ≤10
bgt     t0, t2, error

# input elems of array
la      t3, array

# counter of elems
mv      t4, t0

input_elem_loop:
beqz    t4, sum
la      a0, msg_elem_in
li      a7, 4           # output str
ecall

li      a7, 5           # input int
ecall

sw      a0, 0(t3)       # storege elem in arr
addi    t3, t3, 4       # move to next memory cell +4 byte to ptr
addi    t4, t4, -1      # reducing the counter

la      a0, msg_elem_left
li      a7, 4
ecall
mv      a0, t4          # counter how far to go
li      a7, 1
ecall
la      a0, msg_new_line
li      a7, 4
ecall

j       input_elem_loop


sum:
la      t3, array       # ptr to array start
mv      t4, t0
li      t5, 0           # sum
li      s3, 1           # flag that all is good
```

```
li      s4, 0           # counter of already summed elems
li      s8, 500000000
li      s9, -500000000

sum_loop:
beqz    t4, count_even_odd

lw      t1, 0(t3)       # t1 ptr start arr

add     t2, t5, t1      # set t2 curr sum t5 prev sum
addi    s4, s4, 1

# check overflow
beqz    t5, skip
bgt     t5, s8, check_pos
blt     t5, s9, check_neg
j       sum_valid


skip:
j       sum_valid

check_neg:
bgt     t2, t5, overflow
j       sum_valid

check_pos:
blt     t2, t5, overflow
j       sum_valid

sum_valid:
mv      t5, t2
addi    t3, t3, 4
addi    t4, t4, -1
j       sum_loop

overflow:
li      s3, 0
addi    s4, s4, -1
la      a0, msg_arroverflow
li      a7, 4
ecall

mv      a0, t5
li      a7, 1           # output sum
ecall

la      a0, msg_new_line
li      a7, 4
```

```
ecall

la      a0, msg_curr_count
li      a7, 4
ecall

mv      a0, s4
li      a7, 1           # output counted
ecall

la      a0, msg_new_line
li      a7, 4
ecall

j       count_even_odd

count_even_odd:
la      t3, array

mv      t4, t0          # set count down
li      s0, 0           # counter even
li      s1, 0           # counter odd
li      s2, 2           # divider to check %

count_even_odd_loop:
beqz    t4, logic

lw      t1, 0(t3)
rem     t2, t1, s2
beqz    t2, even_counter
addi    s1, s1, 1
j       next

even_counter:
addi    s0, s0, 1

next:
addi    t4, t4, -1
addi    t3, t3, 4
j       count_even_odd_loop

logic:
beqz s3, output_ans_without_sum         # skip sum output sum cause i'm
bit laze to invent a bicycle

output_ans:
la      a0, msg_sum
li      a7, 4
ecall
```

```
mv      a0, t5
li      a7, 1
ecall

la      a0, msg_new_line
li      a7, 4
ecall

output_ans_without_sum:
la      a0, msg_even
li      a7, 4
ecall

mv      a0, s0
li      a7, 1
ecall

la      a0, msg_new_line
li      a7, 4
ecall

la      a0, msg_odd
li      a7, 4
ecall

mv      a0, s1
li      a7, 1
ecall
j       end

error:
la      a0, msg_error
li      a7, 4
ecall
j       end

end:
# repair s-registr to default value
li      s0, 0
li      s1, 0
li      s2, 0
li      s3, 0
li      s4, 0
li      s8, 0
li      s9, 0
# stop
li      a0, 0
```

```
li      a7, 10
ecall
```