zadanie2.c

```c
#include <stdio.h>
#include <inttypes.h>

long
original_func( char * s, char * d ); // rdi: s, rsi: d
__asm__(
    ".globl original_func\n"
    "original_func:\n"
    "movq %rdi, %rax\n"         // ret = s
    ".myL3:\n"
    "leaq 1(%rax), %r8\n"       // a = ret + 1
    "movb -1(%r8), %r9b\n"      // b = *( a - 1 )
    "movq %rsi, %rdx\n"         // c = d
    ".myL2:\n"
    "incq %rdx\n"               // d ++
    "movb -1(%rdx), %cl\n"      // cl = *( d - 1 )
    "testb %cl, %cl\n"          // cl == 0
    "je .myL7\n"
    "cmpb %cl, %r9b\n"          // cl != b
    "jne .myL2\n"
    "movq %r8, %rax\n"          // ret = a
    "jmp .myL3\n"
    ".myL7:\n"
    "subq %rdi, %rax\n"         // ret -= s
    "ret\n"
);

long
my_func( char * s, char * d ) {
    char * ret = s;
    char * a;
    char b;
    char * c;
    char cl;

    while( 1 ) {
        a = ret + 1;
        b = *( a - 1 );
        c = d;

        do {
            d ++;
            cl = *( d - 1 );
        } while( cl && cl != b );

        if( ! cl ) break;

        ret = a;
    }

    ret -= (long) s;
    return (long) ret;
}
```

```c
long
my_func_clean( char * s, char * d ) {
    long count = 0;

    for( ; *d; s ++ ) {
        while( *d && *d != *s ) d ++;
        if( ! *d ) break;

        count ++;
    }

    return count;
}

int
main( void ) {
    char * a = "abtyfajdfxxadj";
    char * b = "abcdbdefaxx";

    printf( "%ld\n", my_func( a, b ) );
    printf( "%ld\n", my_func( a, a ) );
    printf( "%ld\n\n", my_func( b, b ) );

    printf( "%ld\n", my_func_clean( a, b ) );
    printf( "%ld\n", my_func_clean( a, a ) );
    printf( "%ld\n\n", my_func_clean( b, b ) );

    printf( "%ld\n", original_func( a, b ) );
    printf( "%ld\n", original_func( a, a ) );
    printf( "%ld\n", original_func( b, b ) );

    return 0;
}
```

zadanie3.c

```c
#define A 10
#define B 8

typedef struct {
    int x[A][B];    // 0
    long y;         // 10 * 8 * 4
} str1;

typedef struct {
    char array[B];  // 0
    int t;          // 8
    short s[A];     // 8 + 4 = 12
    long u;         // 12 + 2 * A = 32 => A = 10
} str2;

void
set_val( str1 *p, str2 *q ) { // rdi = p, rsi = q
    long v1 = q->t;     // rax = *( q + 8 )
    long v2 = q->u;     // rax += *( q + 32 )
    p -> y = v1 + v2;   // *( p + 184 ) = rax
}
```

zadanie4.c

```
#define R 7
#define S 5
#define T 13

long A[R][S][T];

long
store_elem( long i, long j,          // rdi = i, rsi = j
        long k, long *dest ) {       // rdx = k, rcx = dest

    *dest = A[ i ][ j ][ k ];        // rax = 3 * j
                                     // rax = 4 * rax + j = ( 3 * j ) * 4 + j
                                     // j = i
                                     // j <<= 6
                                     // i += j
                                     // i += rax
                                     // k += i
                                     // rax = 8 * k + A
                                     // *dest = rax
                                     // rax = 3640 = 455 * 8
                                     // R*S*T = 455
    return sizeof( A );
}
// ( 3 * j ) * 4 + j + i + (i << 6) + k
// (i + i << 6) + 13 * j + k
// 65 * i + 13 * j + k
// (T * S * i) + (T * j) + k

// T = 13
// S = 65 / 13 = 5
// R = 455 / 65 = 7
```

zadanie5.c

```
#define CNT 7
#define SIZE 4

typedef struct {
    long idx;
    long x[ SIZE ];
} a_struct;

typedef struct {
    int first;          // 0
    a_struct a[CNT];    // 8      --- size = 280
    int last;           // 0x120 = 288
} b_struct;

void
test( long i, b_struct *bp ) {          // rdi = i, rsi = bp
    int n = bp -> first + bp -> last;   // ecx = *(bp + 0x120)  -- ecx = bp -> last
                                        // ecx += *bp           -- ecx = bp -> last + bp -> first
    a_struct *ap = & bp -> a[ i ];      // rax = 5 * i
                                        // rax = rax * 8 + bp = bp + ( 5 * i ) * 8
    ap -> x[ ap -> idx ] = n;           // rdx = *( rax + 8 )   -- & bp -> a[ i ]
                                        // rcx = ecx            -- only first bits
                                        // *( rdx * 8 + rax + 16 ) = rcx
}
```

zadanie7.c

```c
#include <stdio.h>

long
switch_prob( long x, long n );  // x: rdi, n: rsi
__asm__(
    ".globl switch_prob\n"
    "switch_prob:\n"
    "sub rsi, 0x3c\n"          // n - 0x3c
    "cmp rsi, 0x5\n"           // n == 5
    "ja this_jump\n"           // rsi > 0x5  --> this_jump
    "jmp []\n"                 // switch( n )
    "lea rax, [rdi * 8]\n"     // ret = x * 8    -- case0, case1
    "ret\n"
    "mov rax, rdi\n"           // ret = x        -- case4
    "sar rax, 3\n"             // ret >>= 3
    "ret\n"
    "mov rax, rdi\n"           // ret = x        -- case2
    "shl rax, 4\n"             // ret <<= 4
    "sub rax, rdi\n"           // ret -= x
    "mov rdi, rax\n"           // x = ret
    "imul rdi, rdi\n"          // x *= x;        -- case5
    "this_jump:\n"             //                -- case3
    "lea rax, [rdi + 0x4b]\n"  // ret = x + 0x4b
    "ret\n"
);

long
my_switch_prob( long x, long n ) {
    long ret;

    n -= 0x3c; // 60
    if( n <= 5 && n != 3 ) {
        switch( n ) {
            case 0:
            case 1: {
                ret = x * 8;
                return ret;
            }
            case 4: {
                ret = x;
                ret >>= 3;
                return ret;
            }
            case 2: {
                ret = x;
                ret <<= 4;
                ret -= x;
                x = ret;
            }
            case 5: {
                x *= ret;
            }
        }
    }

    ret = x + 0x4b; // 75
    return ret;
}
```

```
int
main(void) {

    return 0;
}
```

zadanie8.c

```
#include <stdio.h>

typedef struct A {
    long u[ 2 ];
    long *v;
} SA;

typedef struct B {
    long p[ 2 ];
    long q;
} SB;

SB
eval( SA s );
__asm__(
    ".globl eval\n"
    "eval:\n"
    "mov rax, rdi\n"            // rax = ret structure
    "mov rcx, [rsp + 16]\n"     // rcx = y
    "mov rdx, [rsp + 24]\n"     // rdx = &z
    "mov rsi, [rdx]\n"          // rsi = z
    "mov rdx, rcx\n"            // rdx = y
    "imul rdx, rsi\n"           // y *= z
    "mov [rdi], rdx\n"          // *some_pointer = y
    "mov rdx, [rsp + 8]\n"      // rdx = x
    "mov rdi, rdx\n"            // rdi = x
    "sub rdi, rsi\n"            // some_pointer -= z
    "mov [rax + 8], rdi\n"      // *( some_pointer + 8 ) = some_pointer - z
    "sub rdx, rcx\n"            // x -= orig_y
    "mov [rax + 16], rdx\n"     // *(some_pointer + 16) = y
    "ret\n"
);

SB
my_eval( SA s ) {
    SB ret;

    ret . q      = s . u[ 0 ] - s . u[ 1 ];
    ret . p[ 0 ] = s . u[ 1 ] * *( s . v );
    ret . p[ 1 ] = s . u[ 0 ] - *( s . v );

    return ret;
}
```

```
long
wrap( long x, long y, long z );
__asm__(
    ".globl wrap\n"
    "wrap:\n"
    "sub rsp, 72\n"             // rsp - 72 (???)
    "mov [rsp], rdx\n"          // *rsp = z
    "mov rdx, rsp\n"            // rdx = &z
    "lea rax, [rsp + 8]\n"      // ret = rsp + 8
    "push rdx\n"                // push &z  <-- sub esp, 8
    "push rsi\n"                // push y   <-- sub esp, 8
    "push rdi\n"                // push x   <-- sub esp, 8
    "mov rdi, rax\n"            // x = ret
    "call eval\n"               // eval()
    "mov rax, [rsp + 40]\n"     // ret  = *(rsp + 40)
    "add rax, [rsp + 32]\n"     // ret += *(rsp + 32)
    "imul rax, [rsp + 48]\n"    // rax *= *(rsp + 48)
    "add rsp, 96\n"             // rsp += 96
    "ret\n"                     // return rax
);

long
my_wrap( long x, long y, long z ) {
    long ret;

    SA argument;
    argument . u[ 1 ] = y;
    argument . u[ 0 ] = x;
    argument . v      = &z;

    SB evaled = eval( argument );

    ret = evaled . p[ 1 ];
    ret += evaled . p[ 0 ];
    ret *= evaled . q;

    return ret;
}

int
main(void) {
    printf( "Real wrap: %ld\n", wrap( 58, 22, 56 ) );
    printf( "My wrap:   %ld\n", my_wrap( 58, 22, 56 ) );

    return 0;
}
```