PODPROGRAMY

ZADANIA

Z1. Uruchom i przeanalizuj poniższy program zaimplementowany w języku *C*.

```
#include <stdio.h>
int main(void){
      int H, L, r;
      scanf("%d", &H);
      scanf("%d", &L);
      r = 2*H - L/2;
      printf("%d\n", r);
}
```

Z2. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Python.

```
digits = [0]*99
def int2digits(n, b):
    digits[0] = n % b
    i = 1
    while n > b-1:
        n //= b
        digits[i] = n % b
        i += 1
    return

def length(n, b):
    L=1
    while n > b-1:
        n //= b
        L+= 1
    return L

x = int(input())
base = int(input())
int2digits(x, base)
j = length(x, base) - 1
while j >= 0:
    print(digits[j], end="")
    j -= 1
```

Z3. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Python.

```
Top= 99
Stack= [0]*100
def push(e):
    global Top
    Top -= 1
    Stack[Top] = e
    return
def pop():
    global Top
    e= Stack[Top]
    Top += 1
    return e
x= int(input())
y= int(input())
push(x)
push(y)
print(pop())
print(pop())
```

Z4. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Assembler.

```
%include "AuxMacros.asm"
section .text
global _start
_start:
 mov ax, 3
 mov bx, 7
 push ax
 push bx
 pop ax
 pop bx
 printReg ax
 return0
section .data
HexDig db '0', '1', '2', '3'
 db '4', '5', '6', '7'
 db '8', '9', 'A', 'B'
 db 'C', 'D', 'E', 'F'
msg db '12 = 0000', 0xa
len equ $ - msg
```

Z5. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Assembler.

```
%include "AuxMacros.asm"
section .text
global _start
_start:
 mov ax, 3
 mov bx, 7
 push bx
 mov bx, ax
 add ax, ax
 add ax, ax
 sub ax, bx
 pop bx
 printReg bx
 return0
section .data
HexDig db '0', '1', '2', '3'
 db '4', '5', '6', '7'
 db '8', '9', 'A', 'B'
 db 'C', 'D', 'E', 'F'
msg db '12 = 0000', 0xa
len equ $ - msg
```

Z6. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Assembler.

```
%include "AuxMacros.asm"
section .text
global _start
_start:
 mov ax, sp
 mov bx, 1
 push bx
 sub ax, sp
 printReg ax
 return0
section .data
HexDig db '0', '1', '2', '3'
 db '4', '5', '6', '7'
 db '8', '9', 'A', 'B'
 db 'C', 'D', 'E', 'F'
msg db '12 = 0000', 0xa
len equ $ - msg
```

Z7. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Assembler.

```
%include "AuxMacros.asm"
section .text
global _start
_leng: push dx ; tmp
       push si ; tmp
       push ax ; param n
       push bx ; param b
       mov cx, 1 ; L= 1
_loop: mov si, bx
       sub si, 1
       cmp ax, si
       jng _pool ; while n > b-1:
       mov dx, 0
       div bx ; n //= b
       add cx, 1 ; L += 1
       jmp _loop
_pool: pop bx ; param b
       pop ax ; param n
       pop si ; tmp
       pop dx ; tmp
       ret
_start:
 mov ax, 65
 mov bx, 10
 call _leng
 printReg cx
 mov bx, 8
 call _leng
 printReg cx
 return0
section .data
HexDig db '0', '1', '2', '3'
 db '4', '5', '6', '7'
 db '8', '9', 'A', 'B'
 db 'C', 'D', 'E', 'F'
msg db '12 = 0000', 0xa
len equ $ - msg
```

Z8. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Python.

```
def fact(n):
    prod= 1
    j= 2
    while j <= n:
        prod *= j
        j += 1
    return prod
print(fact(4))
```

Z9. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Python.

```
def fact(n):
    if n==0:
        return 1
    else:
        return n*fact(n-1)
print(fact(4))
```

Z10. Uruchom i przeanalizuj poniższy program zaimplementowany w języku C.

```
#include <stdio.h>
int main(void){
    int A, B, *Adr;
    A=3;
    B=4;
    Adr= &A;    // *Adr == A
    B= B + *Adr;
    printf("%d %d\n", A, B);
}
```

Z11. Uruchom i przeanalizuj poniższy program zaimplementowany w języku Assembler.

```
%include "AuxMacros.asm"

section .text
global _start

fact: push dx    ; tmp
      push ax    ; param n
      cmp  ax, 0
      jne  fi    ; if n==0:
      mov  bx, 1
      pop  ax    ; param n
      pop  dx    ; tmp
      ret        ; return 1
fi:   push ax
      sub  ax, 1
      call fact  ; bx= fact(n-1)
      pop  ax
      mul  bx
      mov  bx, ax; bx= n*bx
      pop  ax    ; param n
      pop  dx    ; tmp
      ret        ; return bx
_start:
       mov  ax, 4
       call fact
       printReg bx ; print(fact(4))
       return0
section     .data
HexDig      db '0', '1', '2', '3'
      db '4', '5', '6', '7'
      db '8', '9', 'A', 'B'
      db 'C', 'D', 'E', 'F'
msg   db '12 = 0000', 0xa
len     equ $ - msg
```

Z12. Uruchom i przeanalizuj poniższy program zaimplementowany w języku C.

```
#include <stdio.h>
void swap1(int a, int b){
     int  tmp;
     tmp= a;
     a= b;
     b= tmp;
     return;
}
int main(void){
     int X, Y;
     X= 1;  Y= 2;
     printf("%d %d; ", X, Y);
     swap1(X, Y);
     printf("%d %d\n", X, Y);
}
```

Z13. Uruchom i przeanalizuj poniższy program zaimplementowany w języku C.

```
#include <stdio.h>
void swap1(int *a, int *b){
     int  tmp;
     tmp= *a;
     *a= *b;
     *b= tmp;
     return;
}
int main(void){
     int X, Y;
     X= 1;  Y= 2;
     printf("%d %d\n", X, Y);
     swap1(&X, &Y);
     printf("%d %d\n", X, Y);
}
```

Z14. Uruchom i przeanalizuj poniższy program zaimplementowany w języku C.

```
#include <stdio.h>
void swap(int a[]){
     int  tmp;
     tmp = a[0];
     a[0]= a[1];
     a[1]= tmp;
     return;
}
int main(void){
     int X[3];
     X[0]= 1;
     X[1]= 2;
     printf("%d %d\n", X[0],X[1]);
     swap(X);
     printf("%d %d\n", X[0],X[1]);
}
```

Z15. Uruchom i przeanalizuj poniższy program zaimplementowany w języku C.

```
#include <stdio.h>
int Sigma(int From, int To, int (*F)(int n)){
    int j, Sum;
    Sum= 0;
    j= From;
    while (j<=To){
       Sum+= F(j);
       j+= 1; }
    return Sum;
}
int fact(int n){
    if (n==0)
       return 1;
    return n*fact(n-1);
}
int main(void){
    printf("%d\n", Sigma(1, 4, fact));
}
```