5. Napisz funkcję, która otrzymuje dwa argumenty: dodatnią liczbę całkowitą

n oraz n-elementową tablicę `tab` o elementach typu `int`. Funkcja ma potroić wszystkie elementy parzyste w tablicy przekazanej jako argument. Stwórz dwa przypadki testowe dla funkcji.

Punktacja: 12 pkt.

```c
#include <stdio.h>

void foo(int n,int tab[n])
{

    for(int i=0;i<n;i++)
    {
        if(tab[i]%2 == 0)
        {
            tab[i]=3*tab[i];
        }
    }
}


int main()
{
   int tab[5]={1,2,3,4,5};
   foo(5,tab);
   for(int i=0;i<5;i++)
   {
       printf("%d\n",tab[i]);
   }

}
```

3. Napisz funkcję, która ma dwa argumenty. Pierwszym argumentem jest wskaźnik na funkcję o jednym argumencie typu `int` zwracającą wartość typu `int`. Drugim argumentem jest wartość typu `int`. Funkcja zwraca wynik dzielenia przez 5 wartości funkcji otrzymanej w pierwszym argumencie na liczbie całkowitej podanej w drugim argumencie. Stwórz przypadek testowy.

```c
#include <stdio.h>



int foo(int (*wsk)(int a), int n)
{
return wsk(n);
}
int foo2(int a)
{
    return a/5;
}

int main()
{
    printf("%d",foo(foo2,10));

}
```

7^(-n),nieujemna, podpowiedz 1/7^n

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


int potega(unsigned int n)
{
    float wynik=1;
    for (int i=0;i<n;i++)
        {
        wynik=wynik*7;
        }
     wynik=1/wynik;
    printf("%f", wynik);
    return 0;
}

int main()
{
    unsigned int n=2;
    potega(n);
    return 0;
}
```

2. Napisz funkcję, której argumentem jest dodatnia liczba całkowita n. Funkcja ma wyświetlić w kolejnych wierszach wszystkie dzielniki n mniejsze niż n. Stwórz przypadek testowy.

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

int foo(int n)
{
      for(int i=1; i<n; i++)
      {
        if(n%i == 0)
        {
            printf("%d\n",i);
        }
      }
}
int main()
{
     int n=20;
     printf("%d",foo(n));
      return 0;
}
```

2. Napisz funkcję, która dostaje jako argumenty liczbę dodatnią n i zwraca jako wartość elementy ciągu: $(2, 6, 18, 54, 162, \ldots)$. Stwórz przypadek testowy.

Punktacja: 9 pkt.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int foo(int n)
{
    return 2*pow(3,n);
}

int main()
{
    printf("%d\n",foo(1));
    printf("%d\n",foo(2));
    printf("%d\n",foo(3));
    printf("%d\n",foo(4));
    return 0;
}
```

3. Napisz funkcję `foo`, która ma dwa argumenty. Pierwszym argumentem jest wskaźnik na funkcję f o dwóch argumentach typu `int` zwracającą wartość typu `int`. Drugim argumentem jest wartość n typu `int`. Funkcja `foo` ma zwrócić wartość $f(n,0)+f(0,n)$. Stwórz dwa przypadki testowe.

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

int foo(int (*wsk)(int a, int b), int n)
{
    return wsk(n,0)+wsk(0,n);
}

int f1(int a, int b)
{
    return 2*a+b;
}

int f2(int a, int b)
{
    return a*b;
}

int main()
{
    printf("%d\n",foo(f1,5));
    printf("%d\n",foo(f2,5));
    return 0;
}
```

5. Napisz funkcję, która otrzymuje trzy argumenty: dodatnią liczbę całkowitą n oraz dwie n-elementowe tablice `tab1` i `tab2` o elementach typu `float`. Funkcja ma zwrócić różnicę wartości najmniejszych z tablicy `tab1` i `tab2`. Stwórz przypadek testowy.

Punktacja: 12 pkt.

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

float najmniejsza(int n, float tab[])
{
    float temp=tab[0];
    for(int i=1;i<n;i++)
    {
        if (temp>tab[i])
            temp=tab[i];
    }
    return temp;
}

float foo(int n, float tab1[], float tab2[])
{
    return najmniejsza(n,tab1)-najmniejsza(n,tab2);
}

int main()
{
    float t1[] = {4,5.3,-9.3};
    float t2[] = {-9.2,1.3,8.3};
    printf("%f\n",foo(3,t1,t2));
    return 0;
}
```

2. Napisz funkcję, której argumentem są trzy liczby całkowite. Funkcja ma zwrócić jeden, jeśli wprowadzane argumenty tworzą trójkę pitagorejską oraz zero w pozostałym wypadku. Stwórz przypadek testowy.

Wskazówka: Trójka pitagorejska – trzy liczby całkowite dodatnie a, b, c spełniające tzw. równanie Pitagorasa: $a^2 + b^2 = c^2$.

Punktacja: 9 pkt.

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

int foo(int a,int b, int c)
{
    if (a*a+b*b==c*c)
        return 1;
    if (a*a+c*c==b*b)
        return 1;
    if (c*c+b*b==a*a)
        return 1;
    return 0;
}

int main()
{
    printf("%d\n",foo(3,4,5));
    printf("%d\n",foo(5,3,4));
    printf("%d\n",foo(3,4,6));
    return 0;
}
```

3. Napisz funkcję `foo`, która ma dwa argumenty. Pierwszym argumentem jest wskaźnik `wsk1` na stałą wartość typu `int`, drugim argumentem jest stały wskaźnik `wsk2` na zmienną typu `int`. Funkcja `foo` ma zwrócić liczbę całkowitą zawierającą różnicę wartości wskazywanej przez pierwszy wskaźnik i wartości wskazywanej przez drugi wskaźnik. Stwórz przypadek testowy.

Punktacja: 10 pkt.

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

int foo(const int *wsk1, int*const wsk2)
{
    return *wsk1-*wsk2;
}

int main()
{
    int a=5, b=7;
    printf("%d\n", foo(&a,&b));
    return 0;
}
```

5. Napisz funkcję, która otrzymuje dwa argumenty: dodatnią liczbę całkowitą n oraz n-elementową tablicę `tab` o elementach typu `int`. Funkcja ma zwrócić sumę elementów znajdujących się na parzystych indeksach. Stwórz przypadek testowy.

Punktacja: 12 pkt.

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

int foo(int n, int tab[])
{
    int temp=0;
    for(int i=0;i<n;i+=2)
    {
        temp+=tab[i];
    }
    return temp;
}

int main()
{
    int tablica[]={3,4,-9,1,-2,5};
    printf("%d\n",foo(6,tablica));
    return 0;
}
```