A1. Napisać funkcję

```
int tylkoDodatnie(int* t, int rozmiar, int* d)
```

która przepisuje (tylko) dodatnie elementy tablicy `t` o rozmiarze `rozmiar` do tablicy `d` (zakładamy, że `d` jest dostatecznie duża). Kolejność elementów powinna zostać zachowana, a wynik powinien być liczbą elementów przepisanych.

A2. Napisać funkcję

```
bool znajdzRowne(int* t, int rozmiar)
```

która zwraca `true` wtedy i tylko wtedy, gdy pewien element w tablicy `t` się powtarza.

A3. Napisać funkcję

```
void wspak(int* t, int rozmiar)
```

która odwraca tablicę t o rozmiarze rozmiar (tzn. pierwszy element ma znaleźć się na końcu, drugi na przedostatnim miejscu, itd.)

A4. Napisać funkcję

```
int sumaDzielnikow(int n)
```

która zwraca sumę wszystkich dzielników liczby n.

A5. Napisać funkcję

```
void usunSpacje(char* s, char* t)
```

która umieszcza w tablicy `t` napis `s` z usuniętymi spacjami. Zakładamy, że w `t` jest dostatecznie dużo miejsca.

A6. Napisać funkcję

```
int pierwiastek(int n)
```

która zwraca pierwiastek z liczby `n` zaokrąglony w dół do liczby całkowitej.

A7. Napisać funkcję

```
int sumaRoznych(int* t, int rozmiar)
```

która zwraca sumę **różnych** elementów tablicy `t` o rozmiarze `rozmiar`.

A8. Napisać funkcję

```
int fibo(int n)
```

która zwraca `n`-tą liczbę Fibonacciego (tj. f(0)=f(1)=1, f(n)=f(n-1)+f(n-2) dla n>1).

A9. Napisać funkcję

```
int drugi(int* t, int rozmiar)
```

która zwraca drugi największy element w tablicy `t` o rozmiarze `rozmiar`. W przypadku, gdy są dwa równe największe elementy należy zwrócić jeden z nich. Można założyć, że `rozmiar`>1.

A10. Napisać funkcję

```
void przesun(int *t, int rozmiar)
```

która przesuwa w prawo elementy tablicy `t` o rozmiarze `rozmiar`. Ostatni element powinien znaleźć się na początku.

A11. Napisać funkcję

```
int a(int n)
```

która zwraca sumę liczb pierwszych nie większych niż n. Można założyć, że parametr n jest liczbą dodatnią.

A12. Napisać funkcję

```
bool f(char* s)
```

która zwraca true wtedy i tylko wtedy, gdy każde dwa kolejne znaki w napisie s są różne.

A13. Napisać funkcję

```
double p(double* t, int r)
```

która zwraca medianę elementów tablicy t o długości r, tj. dowolną liczbę x taką, że liczba elementów tablicy t większych od x jest równa liczbie elementów mniejszych od x. Można założyć, że wszystkie elementy t są różne.

A14. Napisać funkcję

```
int b(int n)
```

która zwraca sumę cyfr liczby n w zapisie dziesiętnym. Można założyć, że parametr n jest dodatni.

A15. Napisać funkcję

```
bool g(char* s, int n)
```

która zwraca true wtedy i tylko wtedy, gdy napis s zawiera ciąg co najmniej n kolejnych jednakowych znaków.

A16. Napisać funkcję

```
int r(int* t, int n)
```

która zwraca liczbę różnych liczb pierwszych występujących w tablicy t o rozmiarze n.

A17. Napisać funkcję

```
bool c(int n)
```

która zwraca true wtedy i tylko wtedy, gdy n jest długością przeciwprostokątnej trójkąta prostokątnego o całkowitych bokach, tj. gdy istnieją liczby całkowite dodatnie a, b takie, że n*n=a*a+b*b.

.

A18. Napisać funkcję

```
int h(char* s)
```

która zwraca liczbę różnych znaków występujących w napisie s.

A19. Napisać funkcję

```
int q(int* t, int r)
```

która zwraca wartość występującą w tablicy t o rozmiarze r największą ilość razy. Jeśli jest wiele takich wartości, należy zwrócić największą z nich.

B1. Podać wynik działania programu

```
int main() {
     int l=0;
     for(int x=5; x; x--) {
          for(int y=x; y; y--)
               for(int z=y; z; z--)
                    l++;
          cout << l << endl;
     }
}
```

B2. Podać wynik działania programu. Wyjaśnić, co zwraca funkcja `f` dla dowolnego parametru.

```
int f(char* s) {
     if (*s)
          return 2*f(++s);
     return 1;
}

int main() {
     cout << f(”Ala”) << endl << f(”Eugeniusz”) << endl;
}
```

B3. Podać wynik działania programu

```
int main() {
     int x=8;
     for(int i=0; i<x; ++i) {
          for(int j=0; j<x; j++) {
               if((i+j)%2)
                    cout << j;
               else
                    cout << 'X';
          }
          cout << endl;
     }
}
```

B4. Podać wynik działania programu. Wyjaśnić, co zwraca funkcja `f` dla dowolnego parametru.

```
void f(int n) {
     if(n) {
          int a;
```

```
          if (a=n%10) {
               cout << a;
               f(n-1);
          }
          else
               f(n/10);
     }
}

int main() {
     f(35); cout << endl; f(825); cout << endl; f(1000);
}
```

B5. Podać wynik działania programu

```
int f(int a, int b) {
     cout << a << ',' << b << endl;
     if (a>0)
          return 1 + f(a+b, b-1);
     return 0;
}
int main() {
     cout << f(21, 0) << endl << f(1, 5) << endl << f (30, -10)
<< endl;
}
```

Czy funkcja f zawsze (tj. dla każdych parametrów) zakończy działanie? Odpowiedź uzasadnić.

B6. Podać wynik działania programu

```
int main() {
     int t[10];
     for(int i=0; i<10; i++)
          t[i]=i+1;
     for(int i=9; i; --i)
          for(int j=i-1; j+1; --j)
               t[i] += t[j];
     for(int i=0; i<10; i++)
          cout << t[i] << endl;
}
```

B7. Podać wynik działania programu

```
int main() {
     char* s = "Alicja w krainie czarow";
```

```
	for(char* p=s; *p; p++) {
		cout << *p;
		if (*p==' ')
			cout << p[1];
	}
}
```

B8. Podać wynik działania programu

```
int f(int n, int d) {
	if (n==1)
		return 1;
	if(n%d)
		return f(n,d+1);
	cout << d << ',';
	return d*f(n/d, d);
}
int main() {
	int a = f(420, 2);
	cout << endl << a;
	a = f(77, 2);
	cout << endl << a;
}
```

B9. Podać wynik działania programu

```
int main() {
	int x=7;
	while(x--) {
		for(int y=0; y<=x; ++y)
			cout << y;
		cout << endl;
	}
}
```

B10. Podać wynik działania programu. Wyjaśnić, co zwraca funkcja f dla dowolnego parametru.

```
int f(int n) {
	if (n)
		return n%2 + f(n/2);
	return 0;
}
int main() {
	cout << f(10) << endl << f(31) << endl << f(256) << endl;
}
```