

Programowanie komputerowe 2021/22

Zadania przygotowawcze do kolokwium

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.

A20. Napisać funkcję

```
bool a(int* t, int r)
```

która zwraca `true` wtedy i tylko wtedy, gdy elementy tablicy `t` o rozmiarze `r` tworzą ciąg arytmetyczny.

A21. Napisać funkcję

```
void f(char* s)
```

która drukuje na ekranie tekst zawarty w napisie `s` w którym ciągi kolejnych spacji są zastąpione przez znak `*`. Np. po wywołaniu `f("AB A CC CV ")` powinno zostać wydrukowane `AB*A*CC*CV*`.

A22. Napisać funkcję

```
bool b(int t[], int r)
```

która zwraca `true` wtedy i tylko wtedy, gdy w tablicy `t` o rozmiarze `r` znajdują się dokładnie dwie różne wartości.

A23. Napisać funkcję

```
bool g(char* s, char* t)
```

która zwraca `true` wtedy i tylko wtedy, gdy napis `s` można uzyskać z napisu `t` przez wykreślenie pewnych znaków. Np. `g("ABC", "CADCBC")=true`, `g("ABC", "DADCBD")=false`.

A24. Napisać funkcję

```
bool c(int t[], int r)
```

która zwraca `true` wtedy i tylko wtedy, gdy ciąg różnic kolejnych elementów w tablicy `t` o rozmiarze `r` jest ściśle rosnący. Np. `c({1, 3, 6}, 3)=true` (bo $6-3 > 3-1$).

A25. Napisać funkcję

```
void h(char* s)
```

która drukuje na ekranie wszystkie trzyliterowe wyrazy z napisu `s`. Można założyć, że `s` zawiera tylko litery i spacje. Np. `h("AAA B AAAB AAA CAC")` powinno wydrukować `AAA AAA CAC`.

A26. Napisać funkcję

```
bool a(int t[], int r)
```

która zwraca `true` wtedy i tylko wtedy, gdy żaden element tablicy `t` o rozmiarze `r` nie jest dzielnikiem innego elementu.

A27. Napisać funkcję

```
void f(char* s)
```

która zwraca długość najdłuższego palindromu występującego w napisie `s`. Napis to palindrom

jeśli brzmi tak samo czytany od lewej i od prawej strony. Przykłady: $f("ABCDEDCBA")=5$,
 $f("AACDAA")=2$, $f("ABCDE")=1$.

A28. Napisać funkcję

```
bool b(int t[], int r)
```

która zwraca true wtedy i tylko wtedy, gdy wszystkie elementy w tablicy t o rozmiarze r są różne i dodatnie.

A29. Napisać funkcję

```
void g(char* s)
```

która zastępuje wszystkie wystąpienia ciągu liter "AB" przez gwiazdkę. Przykłady: g zamienia "ABCDEFABBA" na "*CDEF*BA", "BABA" na "B*A" a "CBA" pozostawia bez zmian.

A30. Napisać funkcję

```
int a(int t[], int n)
```

która zwraca sumę tych elementów tablicy t o rozmiarze n , które są większe od obu sąsiadów (lub jednego jeśli jest to skrajny element). Np.

$a(\{1, 2, 3, 2, 5, 1\}, 6)$ powinno zwrócić $8=3+5$,

$a(\{2, 1, 2\}, 3)$ powinno zwrócić $4=2+2$,

$a(\{1, 1, 1\}, 3)$ powinno zwrócić 0.

A31. Napisać funkcję

```
void f(char* s)
```

która drukuje napis s , ale jeśli występuje w nim ciąg kolejnych jednakowych znaków, dany znak jest drukowany tylko raz, np.

$f("AAABBCBAAB")$ powinno wydrukować ABCBAB,

$f("A AAA AAA")$ powinno wydrukować A A A.

A32. Napisać funkcję

```
void b(char* s)
```

która drukuje tylko znaki napisu s mające parzyste indeksy. Np.

$b("ABCDEF")$ powinno wydrukować ACE.

A33. Napisać funkcję

```
int g(int t[], int n)
```

która zwraca sumę elementów tablicy t o rozmiarze n , które występują w niej tylko jeden raz. Np.

$g(\{1, 1, 2, 3, 1\}, 5)$ powinno zwrócić $5=2+3$,

$g(\{4, 4, 4\}, 3)$ powinno zwrócić 0 (bo żadna wartość nie występuje jednokrotnie).

A34. Napisać funkcję

```
void a(char* s)
```

która drukuje pierwsze litery wyrazów w napisie s . Np.

$a("czarny kot")$ drukuje ck,

$a(" zielony las ")$ drukuje zl.

A35. Napisać funkcję

```
bool f(int t, int n)
```

która zwraca true jeśli żaden z elementów tablicy `t` nie jest sumą dwóch innych elementów tej tablicy. Parametr `n` jest rozmiarem tablicy `t`. Np.

```
f({1, 9, 3, 7}, 4)=true,
```

```
f({3, 6, 2, 11, 4})=false (bo 2+4=6).
```

A36. Napisać funkcję

```
void b(char* s)
```

która drukuje pierwszą literę napisu `s` raz, drugą dwa razy, trzecią trzy razy, itd. Np `b("ABC")` drukuje `ABBCCC`.

A37. Napisać funkcję

```
bool g(int* t, int n)
```

która zwraca `true` wtedy i tylko wtedy, gdy każde dwa elementy `t` są względnie pierwsze, tj. nie posiadają wspólnego dzielnika większego niż 1. Parametr `n` jest rozmiarem tablicy `t`. Np.

```
g({5, 13, 24, 111}, 4)=false, bo 3 dzieli zarówno 24 jak i 111,
```

```
g({77, 256, 15}, 3)=true.
```

A38. Napisać funkcję

```
void c(char* s)
```

która drukuje napis `s` z pominięciem początkowych i końcowych znaków `*`, np.

```
c("***A*B*") drukuje A*B,
```

```
c("AB*C**D") drukuje AB*C**D.
```

A39. Napisać funkcję

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

która zwraca najmniejszą liczbę dodatnią `k` taką, że co `k`-ty element tablicy `t` się powtarza.

Parametr `n` jest długością tablicy `t`. Np.

```
h({1, 2, 1, 2, 1, 2, 1}, 7)=2,
```

```
h({1, 5, 3, 7, 1, 5}, 5)=4,
```

```
h({2, 7, 2, 3, 7, 2, 4}, 7)=7.
```

A40. Napisać funkcję

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

która drukuje w pierwszym wierszu `n` gwiazdek, w drugim `n-1` gwiazdek i następnie 1 kropkę, w trzecim `n-2` gwiazdki i następnie 2 kropki, itd. Np. `a(4)` powinno wydrukować

```
****
```

```
***.
```

```
**..
```

```
*...
```

A41. Napisać funkcję

```
bool p(int t[], int n, int d)
```

która zwraca `true` jeśli każde dwa elementy tablicy `t` różnią się o co najmniej `d`. Parametr `n` jest rozmiarem tablicy.

A42. Napisać funkcję

```
void v(char* s)
```

która przesuwa w prawo wszystkie cyfry występujące w napisie `s`. Ostatnia cyfra trafia na miejsce pierwszej cyfry. Znaki, które nie są cyframi pozostają na swoim miejscu. Na przykład:

- jeśli `s="A12B34"`, to po wywołaniu `v(s)`, `s` zamienia się na `"A41B23"`

- jeśli `s="9x001y2"`, to po wywołaniu `v(s)`, `s` zamienia się na `"2x900y1"`

- jeśli `s="AB5C6D"`, to po wywołaniu `v(s)`, `s` zamienia się na `"AB6C5D"`

- jeśli `s="7"`, to po wywołaniu `v(s)`, `s` zamienia się na `"7"`

- jeśli `s="ABC"`, to po wywołaniu `v(s)`, `s` zamienia się na `"ABC"`.

Uwaga: znak `c` jest cyfrą jeśli spełnia warunek `('0'<=c && c<='9')`

A43. Napisać funkcję

```
bool q(int t[], int n)
```

która zwraca `true` jeśli każdy element tablicy jest większy od sumy poprzedzających go elementów. Parametr `n` jest rozmiarem tablicy. Np.

```
q({1, 3, 8, 20}, 4)=true,
```

```
q({3, 7, 11, 20}, 4)=false (bo 3+7+11>=20)
```

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;
}
```

B11. Podać wynik działania programu

```
char f(int n, int k) {
    if(n==k || k==0) return '*';
    if(f(n-1,k-1)==f(n-1,k))
        return '.';
    else
        return '*';
}
int main() {
    for(int i=0; i<=5; i++) {
        for(int j=0; j<=i; j++)
            cout << f(i,j);
        cout << endl;
    }
}
```

B12. Podać wynik działania programu

```
void a(int n) {
```

```

        if(n==0)
            cout << '.';
        else {
            a(n-1); a(n-1); a(n-1);
        }
    }
void b(int n) {
    if(n==0) cout << '*';
    else {
        b(n-1); a(n-1); b(n-1);
    }
}
int main() {
    b(3);
}

```

B13. Podać wynik działania programu

```

void g(int n) {
    if(n%2==0) {
        cout << '.';
        g(n/2);
    }
    else cout << '*';
}
int main() {
    for(int i=250; i<=260; i++)
        g(i);
}

```

B14. Podać wynik działania programu

```

void a(int m, int n) {
    int a=m; int b=n; int c; int d;
    do {
        c=-b; d=a+b;
        a=c; b=d;
        cout << a << "," << b << ";";
    } while(a!=m || b!=n);
    cout << endl;
}
int main() {
    a(1,-1); a(3,4);
}

```

B15. Podać wynik działania programu

```
char a(int n, int k){
    if(k<0 || k>n) return 'Q';
    if(k==0 || k==n) return '+';
    if(a(n-1,k-1)==a(n-1,k)) return ' ';
    return '#';
}
int main(){
    for(int i=0; i<5; i++) {
        for(int j=0; j<5; j++)
            cout << a(i,j);
        cout << endl;
    }
}
```

B16. Podać wynik działania programu

```
void a(int n) {
    int m=n;
    do {
        n=m;
        m=(n+11)/3; cout << n << " ";
    } while(n!=m);
    cout << endl;
}
int main() {
    a(6); a(1); a(60);
}
```

B17. Podać wynik działania programu

```
void q(int& a, int& b, int& c) {
    cout << a << " " << b << " " << c << endl;
    int x=a; int y=b; int z=c;
    a=y+z; b=z+x; c=x+y;
}
int main() {
    int d=2; int e=1; int f=0;
    for(int i=0; i<5; i++)
        q(d,e,f);
}
```