Z0. (4 p.) Zapisz dokładnie, co pojawi się na ekranie.

```
int i;
for (i=11; i>=0; i=i-2)
cout << i << " " << i;
cout << i;
```

Z1. (4 p.) Oblicz końcową wartość *licz*.

```
int licz=1;
for (int i=0; i<2019; i=i+1)
{
  if (i%10>0)
    licz=licz+1;
  else
    licz=licz-10;
}
```

Z2. (4 p.) Ile razy wyświetlą się poszczególne cyfry?

```
for (int i=0; i<=99; i=i+1)
{
  if (i%10==0)
  cout << 1;
  else
  if (i%2==1)
  cout << 2;
  else
  cout << 3;
}
```

* Jaką drobną rzecz wystarczy zmienić w ciele pętli, żeby dwójek wyświetlało się dwa razy mniej niż teraz (a jedynek i trójek wszystko jedno ile)?

II termin

0. (2 p.) Co się wyświetli?

```
for (int i=66; i>=0; i--)
cout << i/7;
```

1. (3 p.) Jaka będzie wartość *i*, a jaka *s* po wyjściu z pętli?

```
int i, s=0;
for (i=1000; i>0; i=i-2)
if (i%10>0) s++;
```

6*. Zapisz w C++ algorytm, który obliczy 1!+2!+3!+...+44!

2. (4 p.) Ile razy wyświetlą się poszczególne cyfry?

```
for (int i=0; i<2019; i++)
{
  if (i%2==0)
  if (i%3==0)
  cout << 1;
  else
  cout << 2;
}
```

III termin

Z0. (3 p.) Oblicz końcową wartość *licz*.

```
int licz=123;
for (int i=1000; i<2020; i=i+2)
{
  if (i%20==0)
    licz=licz+10;
  else
    licz=licz-1;
}
```

Z1. (5 p.) Ile razy wyświetlą się poszczególne cyfry?

```
for (int i=99; i>0; i--)
{
  if (i%2>0)
  cout << 1;
  else
  if (i%4==1)
  cout << 2;
  else
  cout << 3;
  if (i%4==2)
  cout << 4;
}
```

Z2. (5 p.) Zapisz fragment programu obliczający wartość $2^2–3^3+5^2–6^3+…$ aż do 2019 w odpowiedniej potędze.

0. (3+4 p.) Zapisz dokładnie wszystko, co się wyświetli.

a)
```
for (int i=1; i<=5; i++)
for (int j=i; j>0; j--)
cout << i%j;
```

b)
```
for (int i=100; i>=0; i--)
for (int j=10; j<i; j=j+10)
if (i%j==0)
cout << i << j << endl;
```

Z1. (4 p.) Zapisz fragment programu obliczający wartość $-1+5^2–9+13^2–17$ itd. aż do 1001 w odpowiedniej potędze.

Z*. Dla jakich *n* ze zbioru {77, 88, 99, 100, 101, 111} po danej pętli *k* będzie równe 4? Uzasadnij! (Podpowiedź/przypomnienie: 101 jest liczbą pierwszą).

```
int k=0;
for (int i=1; i<=n; i++)
if (n%i==0) k++;
```

II termin (3×1 pkt)

0. Zapisz dokładnie wszystko, co się wyświetli.

a)
```
for (int i=1; i<9; i++)
{
for (int j=i; j>0; j--)
cout << i;
for (int j=0; j<i; j++)
cout << j;
cout << endl;
}
```

b)
```
for (int i=44; i>=0; i=i-4)
for (int j=0; j*j<i; j=j+10)
cout << j << endl;
```

1. Jaka będzie wartość *k* po wyjściu z pętli?

```
int k=0;
for (int i=0; i<=2019; i=i+11)
if (i%2<1)
if (i%11==0)
k++;
```

6*. Nie używając słowa *while*, zapisz w C++ algorytm znajdujący największą potęgę dwójki dzielącą daną liczbę całk. dod. *n* (czyli np. odpowiedziami dla *n* = 1, 4, 5, 18 są kolejno: 1, 4, 1, 2).

```
for (int i=9; i>0; i--) for (int j=i%2; j<i; j=j+2) cout << j;
```

0. (3 p.) Co wyświetli powyższy fragment programu?
1. Zapisz fragment programu, który dla podanego całkowitego dodatniego *n*:
 a) (3 p.) wyświetli trójkąt, który dla *n* = 5 wygląda jak ten po prawej;
 b) (4 p.) obliczy sumę *n* początkowych wyrazów ciągu: $^1/_2$, $^3/_4$, $^5/_8$, $^7/_{16}$, ... (Jeśli to za trudne, możesz – za maks. 2,5 pkt! – obliczyć analogiczną sumę, gdzie wszystkie składniki mają w liczniku 1);
 C*) znajdzie wykładnik największej potęgi dziesiątki nieprzekraczającej *n*, czyli dla *n* = 1, 99, 123 wynikami powinny być odpowiednio 0, 1 i 2. Uwaga: na 6 należy zrobić to bez użycia słowa *while*!

```
1
21
321
4321
54321
```

II termin

```
int i,j; for (i=9; i<13; i++) { for (j=i; j>i%2; j=j-2) cout << i; cout << j; }
```

0. (3 p.) Co wyświetli powyższy fragment programu?
1. (4 + 4 p.) Zapisz fragment programu, który dla podanego całkowitego dodatniego *n*:
 a) wyświetli trójkąt, który dla *n* = 5 wygląda jak ten po prawej
 (wsk.: tam, gdzie nic nie widać, komputer może wypisywać…);
 za mniej pktów możesz sobie coś ułatwić, ale napisz, jaki trójkąt powstanie w Twoim programie!
 b) obliczy wartość $^3/_n - {}^9/_{n-1} + {}^{27}/_{n-2} - {}^{81}/_{n-3}$ itd., aż mianownikiem będzie 1;
 C*) znajdzie NWD dwóch liczb, nie korzystając z żadnych specjalnych algorytmów
 (wsk.: można skorzystać z definicji – *największy wspólny dzielnik*).

```
54321
4321
321
21
1
```

Zapisz w C++ (3 × 4 pkt):

x) fragment programu równoważny fragmentowi
`for (i=0; i<77; i++) { for (j=1; j<10; j++) k=k+j; n=n+k; }` ,
nie używając pętli innych niż *while*;

y) procedurę, która dla argumentów *w* i *k* wypisze *k* początkowych kolumn
w początkowych wierszy tabliczki mnożenia (czyli dla *w*=2 i *k*=3 zobaczymy to:
(Spokojnie załóż, że *w* ani *k* nie będą spoza zbioru {1, 2, 3, …, 9}.);

```
1 2 3
2 4 6
```

z) funkcję obliczającą dla podanej jej jako argument całkowitej dodatniej liczby *n* wartość $1/1 + 1/2^2 + 1/3^3 + \ldots + 1/n^n$.

*. Nie obliczając NWD, zdefiniuj w C++ funkcję NWW o dowolnej złożoności.
(Podp.: można (ale nie trzeba) ustalić jej wartość z definicji – jest to wszak *najmniejsza wspólna wielokrotność*).

(po 4 pkt)
α) Jakie będą *i* i *j* po wyjściu z pętli przy *n* = 1? 2? 3? 4? 5? 11?

```
i=1; j=1;
while (i<n and i%5<4)
{
   i=i+j;
   j++;
}
```

β) Zapisz (w C++!) procedurę, która wypisze w kolejności malejącej 123 początkowe wielokrotności liczby 14 mniejsze od podanego *n*. (Czyli np. dla *n* = 33 powinno się wypisywać 28, 14, 0, -14 itd.).

γ) Napisz funkcję, która odpowie, czy jej argument (liczba całk. dodatnia) ma więcej niż 14 dzielników. Na maks. ocenę funkcja nie powinna mieć zbyt dużej złożoności (więc np. po znalezieniu 14 dzielników…).

* = γ, ale pytamy o czynniki pierwsze.

0. (4 × 2 pkt) Oblicz, zapisując rozwiązanie tak, żebym wiedział, że wiesz:

$x = 2^{100}$ shl 3, $y = 9$ shr 2, $z = (2^{99}+1)$ and $(2^{99}+2)$ and $(2^{99}+3)$ and … and $(2^{99}+15)$,

ś = 2019 xor 1234 xor 2019 xor 1234 xor … xor 2019, gdzie 2019 występuje 100 razy

1. (3 × 2 pkt) Zapisz, czy równanie/nierówność ma zero, jedno czy więcej rozwiązań. Jeśli ma zero – krótko uzasadnij, jeśli jedno – podaj je, jeśli więcej – podaj dowolne dwa.

α) 25 or $x > 25$, β) 40 and $x = 40$, γ) 1234567 xor $x = 1$

6*. Ile rozwiązań mniejszych od 256 ma równanie x or 91 = 91 ?

0. (4 × 2 pkt) Oblicz, zapisując rozwiązanie tak, żebym wiedział, że wiesz:

$x = 2^{99}$ shl 2, $y = 13$ shr 3, $z = (2^{100}+1)$ and $(2^{100}+2)$ and $(2^{100}+3)$ and … and $(2^{100}+15)$

ś = 4321 xor 2020 xor 4321 xor 2020 xor … xor 4321, gdzie 2020 występuje 99 razy.

1. (3 × 2 pkt) Zapisz, czy równanie/nierówność ma zero, jedno czy więcej rozwiązań. Jeśli ma zero – krótko uzasadnij, jeśli jedno – podaj je, jeśli więcej – podaj dowolne dwa.

α) 21 and $x = 16$, β) 1234567 or $x < 1234567$, γ) 123 xor $x = 2^{99}$

6*. Ile rozwiązań mniejszych od 256 ma równanie x or 124 = 124 ?

x) (2+4+2 p.) W $\mathbf{Z}_{201}$ podaj elementy przeciwne i odwrotne (jeśli nie istnieją, to wyraźnie to napisz) do:
$a = 0$, $q = 1$, $b = 2$, $c = 200$, $ć = 55$. Przy odwrotnościach lub ich nieistnieniu chcę widzieć uzasadnienia odpowiedzi!

y) (2 p.) Możliwe sprytnie oblicz 4221·4334 mod 4321.

Z) (2 p.) Czy 25 jest odwracalne w $\mathbf{Z}_{123456789}$? Uzasadnij! (Uzasadnienie nie powinno wymagać zbyt wielu rachunków!)

Z*) Czy istnieją takie całkowite α i β, że $2 = \alpha \cdot 5^{100} + \beta \cdot 123456789$?

0. (3 p.) Może nie wiedziałaś/-łeś, ale (222111·444) mod 366307 = 1!
Czy liczba 222 jest odwracalna w $\mathbf{Z}_{366307}$, a jeśli tak, to jaką ma odwrotność?

1. W $\mathbf{Z}_{303}$:

a) (2 p.) oblicz: $x = 297 \cdot 298 \cdot 299$;

b) (2+4+2 p.) znajdź (jeśli się da) elementy przeciwne i odwrotne do:
0, 1, 300, 302, 202, 152, 55. Odwrotności i ich nieistnienia uzasadnij!

6. (3 p.) Udowodnij, że wśród reszt z dzielenia przez 2020 liczb $1^2, 2^2, 3^2, \ldots, 2019^2$ jest najwyżej 1010 wartości.

```
#include <iostream>

using namespace std;

int fun(int x, int y) {
    int m=x%2;
    int i=m+2;
    for (i; i*i<=x; i++)
    if (y%i==0) return 2; else return 1;
    return x;
    return i;
}

int main() {
    int x,y,m;
    cin >> x >> y;
    cout << fun(x,y);
    return 0;
}
```

0. (5 p.) Zapisz (w C++) funkcję, która obliczy sumę cyfr swojego argumentu.

1a. (2 p.) Które fragmenty zaprezentowanego kodu można usunąć bez wpływu na widoczny efekt jego działania? Krótko uzasadniaj!

b. (3 p.) Po wczytaniu jakich dodatnich x i y program wypisze 2? Uzasadnij!

```
int t[3]={3,4,5}, x=7, a=11, b;

void qq(int a, int &b)
{
    a=b;
    b=x;
}
```

```
void qwerty(int ttt[3], int &a)
{
    ttt[1]=a;
    a=ttt[2];
}
```

Przed mainem zapisano wszystko powyższe. Jak zmienią się poszczególne zmienne globalne, jeśli w mainie wykona się (tylko) dana instrukcja? Skrótowo / symbolicznie uzasadniaj!

```
1) qq(a,b);     3) qq(a,a);          5) qwerty(t,x);
2) qq(x,a);     4) qq(t[1],t[2]);    6) qwerty(t,t[1]);
```

Z0. (4 p.) Zaprezentuj jednoczesny wybór min. i maks. tablicy (8, 4, 3, 2, 2, 6, 9, 5, 1, 1).
Z2. (5 p.) Dlaczego aktualizacji (pozycji) minimum w sortowaniu przez proste wybieranie nie może być więcej niż $n(n-1)/2$?
Z4. (6 p.) Zapisz (w C++) algorytm sortowania przez wybór maksimum.
Z6. (5 p.) Czy istnieją takie dane, których sortowanie przez proste wybieranie działa szybciej niż kwadratowo? Nie musisz wykonywać rachunków, ale zapisz, co trzeba by wyliczyć, i wynik, który decyduje o odpowiedzi.

S1. Zapisz algorytm jednoczesnego wyboru min. i maks. Ustal liczby porównań i podstawień (minimalne, maksymalne, średnie).
S2. Ustal minimalne liczby porównań i podstawień w sortowaniu przez prosty wybór.
S3. Ustal liczby porównań i podstawień w select sorcie dla antyposortowanej tablicy.
S4. Czy sortowanie przez wybieranie jest stabilne? A bąbelkowe?

0. Ile zamian i ile porównań wykona się przy sortowaniu ciągu (1, 2, 3, 4, 9, 0, 5, 6, 7, 8) przez prostą zamianę? Opisz dokładnie, jak wpłyną na te liczby dwa wybrane przez Ciebie ulepszenia. UZASADNIAJ!

0. (4) Scalamy dwa fragmenty tablicy (2, 4, 5, 7, 7, 8, 0, 1, 3, 4, 6). Zapisz, jak zmieniają się użyte zmienne.

1. (8 p.) Sortujemy przez złączanie tablicę 33-elementową x) zstępująco; y) wstępująco. W obu wariantach podaj pierwsze 3 i ostatnie 3 wywołania procedury *Merge*. Ile poziomów (wraz z inicjującym całe sortowanie) ma drzewo wywołań w rozwiązaniu rekurencyjnym? Naszkicuj taki jego fragment, żeby było widać odpowiedź.

6. (6 p.) Oszacuj złożoność quick sorta, jeśli za każdym razem pivotem okazała się najmniejsza wartość w danym fragmencie tablicy. Na 5(+) opisz chociaż działanie QS w takim przypadku, ustal, ile minimów będzie musiało być znalezionych, ile będzie wywołań…

0. Zanalizuj dokładnie (podając przebieg podziału i kolejne wywołania) przebieg QS w wersji Hoare'a na tablicy (4, 2, 1, 3, 5).
2. Ile minimalnie poziomów ma drzewo wywołań QS w wersji Hoare'a dla x) 127, y) 128, z) 129 liczb? Uzasadnij np. jakimś szkicem, podając/opisując przykładowe dane, dla których tak to przebiegnie.
4. Czemu jako pivota lepiej nie wybierać ostatniego elementu?
6. Podaj przewagę MS nad QS i odwrotnie.

0. Zapisz (krótko uzasadniając) wyrażenie C++, którego wartość jest [pseudo]losową czterocyfrową liczbą parzystą.
1. Napisz funkcję *wporzo*, która stwierdzi, czy podane jej dwa stringi (załóżmy dla uproszczenia, że złożone wyłącznie z liter) są w porządku alfabetycznym. (Tzn. np. *wporzo*("INFORMATYKA","MATMA")=1).
2. Zapisz optymalny algorytm sprawdzania palindromiczności napisu złożonego z samych małych liter. Jaką dokładnie ma złożoność optymistyczną, a jaką pesymistyczną? Dla jakich ciągów zachodzi sytuacja pesymistyczna?

0. (8 p.) Ustal dokładnie i z uzasadnieniem optymistyczną i pesymistyczną złożoność algorytmu sprawdzającego, czy dwa napisy długości *n* są anagramami: 0.0 przy pomocy dwóch histogramów,
0.1 przy pomocy jednego.
1. (1+3 p.) Jaki szyfr polialfabetyczny znasz? Co to jest szyfr polialfabetyczny?
2. (6 p.) Zapisz funkcję *Cezar* działającą na napisach złożonych z dużych liter angielskich z dowolnym całkowitym przesunięciem (np. -50), no, takim, że nie grozi nam wyskok poza zakres chara.

Z0. *M* jest macierzą sąsiedztwa grafu *G* o 13 wierzchołkach, niekoniecznie prostego.
Zapisz w C++ funkcję, która stwierdzi, czy jest jednobieżny.

Z1. Ile jest liczb, których zapis trójkowy ma 11 cyfr i jest palindromem?
Jak wyglądają najmniejsza i największa z nich i jakie to wartości?

Z2. Ruch w grze „Samotnicz-k” polega na tym, że daną liczbę naturalną *n* mnoży się przez 5, a następnie do wyniku dodaje 1, 2, 3 albo 4. Jakie liczby da się uzyskać, startując z zera? Jak możliwie najprościej obliczyć największą wartość, jaką da się wtedy uzyskać po 100 ruchach? A najmniejszą?

Z3. Startujemy z (0, 0) i możemy poruszać się o jednostkę w poziomie (za co płacimy 1 zł) i o jednostkę w pionie (za co płacimy 5 zł). Na ruchy wydaliśmy równo 2021 zł.
Czy możemy znajdować się w: $A = (1, 111)$, $B = (1, 222)$, $C = (2, 111)$, $D = (2, 222)$?
W ilu punktach możemy być, jeśli w żadnym ruchu nie poszliśmy w lewo ani w dół?

0. Czy przy sprawdzaniu anagramowości dwoma histogramami lub jednym ma znaczenie, który napis rozpatrujemy jako pierwszy? Uzasadnij odpowiedź dla obu algorytmów.
1. Jaki typ szyfru reprezentuje szyfr Cezara? Użyj dwóch przymiotników, uzasadniając.
2. Działamy na dużych literach alfabetu angielskiego. Mając daną funkcję *Vigenere*, użyj jej do definicji funkcji *Cezar* i *deCezar* z dowolnym dodatnim przesunięciem.

GR. 2:

0. Napisz funkcję, która odwraca podany jako argument napis (czyli np. z „ANETA” powinna wyjść „ATENA”); zrób to, zmieniając tylko daną zmienną, bez żadnych pomocniczych!

1. Ile razy zostanie sprawdzony warunek w ifie, gdy nastąpi wywołanie:
   a) f("AAABAAABC","BC");
   b) f("AAABAAABC","AAAA");
   c) f("AAABAAABC","ABAA") ?
1’. Napisz program, który będzie automatycznie odpowiadać na to pytanie.
* Popraw tę funkcję, tak żeby sprawdzeń tego warunku było możliwie najmniej.

```
int f(string n, string w)
{
   for(int i=0; i+w.size()<=n.size(); i++)
      for(int j=0; j<w.size() and n[j+i]==w[j]; j++)
         if(j+1==w.size())
            return i;
   return -1;
}
```

1. Co to jest szyfr przestawieniowy?
2. Zapisując skrótowo/symbolicznie, co robisz, zapisz wiadomość „INFORMATYKA” szyfrem:
x) GAWERYPOLUKI, y) płotkowym o wys. 5, z) Vigenère’a o kluczu „BYX”.
3. Jaka wiadomość po zaszyfrowaniu płotkiem o wys. 3 da napis „INFORMATYKA”? Pokaż, skąd wiesz!
4. Zapisz w C++ definicję (klasycznej) funkcji Cezar przez odpowiednie wywołanie funkcji Vigenere.
Na 6 dla Rozszerzonych: to samo, ale Cezar z dowolnym przes. z [1, 26].

kl. III

0. (6 p.) Zapisz kolejno wykonywane scalenia i – tam, gdzie ma to sens – wywołania procedury sortującej przy zastosowaniu merge sorta do 7-elementowej tablicy: x) wstępująco, y) zstępująco.
Co można uznać w tych algorytmach za operację elementarną? Ile wykona się ich optymistycznie i ile pesymistycznie w algorytmie x, a ile w y?

1. (4 p.) Jakie będą kolejne wywołania naszej procedury QuickSort i jakie pivoty w każdym z nich przy sortowaniu tablicy (4, 5, 1, 5, 6, 3, 2)?

I dalej mam bałagan, więc kl. IV:

0. (4 p.) Zaprezentuj postać tablicy/wektora w kolejnych krokach uproszczonego algorytmu pakowania plecaka o pojemności 14, jeśli kolejno rozpatrywane są przedmioty o nast. parametrach (objętość, wartość): (3, 12), dwa (5, 8) i dwa (4, 11). Ustal, co jest rozwiązaniem problemu (która liczba i co znaczy).

1. Zapisz (w C++) procedurę, która dla podanego jej jako argument:

x) (3 p.) wypełnionego dynamicznie w uproszczonym algorytmie pakowania plecaka wektora odpowiedzi (czyli takiego, jaki miałxś uzyskać ręcznie w poprzednim zadaniu, ale procedura ma działać niezależnie od pojemności plecaka) poda (po polsku, pełnym zdaniem) rozwiązanie problemu.

y) (5 p.) wektora par złożonych (w tej kolejności!) z nazwiska człowieka i liczby lat, które przeżył, ułoży je od najstarszych do najmłodszych, a osoby w tym samym wieku – alfabetycznie nazwiskami.

Z*)rób to (y) na dwa sposoby!

(2+6 p.) Xo to jest algorytm zachłanny? Opisz dwa zachłanne podejścia do zagadnienia plecakowego i stwierdź o każdym (z uzasadnieniem!), czy jest jego rozwiązaniem.

(10 p.) Yaką strukturą jest `vector <int> t[4]` ? Zapisz, jak do jej sortowania użyć funkcji `sort`, i opisz, jaki będzie efekt jej działania. Możesz użyć odpowiedniego przykładu.

Aby celowo zaburzyć porządek alfabetyczny (np. żeby oddalić od siebie wyrazy pokrewne (jak „informatyka”, „informatyk”, „informatyczka”, „informatyczny”, …) albo rózne formy tego samego wyrazu (jak „matematyk”, „matematyka”, „matematykowi”, „matematykiem”, …)), językoznawcy stosują nieraz słowniki „a tergo” (co znaczy po łacinie ‘od tyłu’), czyli spisy wyrazów w takiej kolejności, w jakiej ułożyłyby się alfabetycznie, gdyby napisano je od tyłu. Np. uporządkowane a tergo są wyrazy *żaba, człowiek, pies, kot, but*.

(22 p.) Zapisz (w C++) procedurę, która dla podanego jako argument wektora liczb dwucyfrowych uporządkuje je a tergo: 0) bez użycia niestandardowych bibliotek, 1) przez zastosowanie funkcji `sort`.