Organizatorzy:
Polskie Towarzystwo Informatyczne Oddział Kujawsko-Pomorski
Uniwersytet Mikołaja Kopernika w Toruniu Wydział Matematyki i Informatyki
Centrum Kształcenia Ustawicznego TODMiDN w Toruniu

SCHEMAT OCENIANIA
poziom rozszerzony
arkusz I

UWAGA: Za prawidłowe rozwiązania inne niż w kluczu przyznajemy maksymalną liczbę punktów.

Zadanie 1 — TEST (5 pkt.)

Numer zadania	Część zadania	Prawidłowa odpowiedź	Maksymalna punktacja za czynność	Maksymalna punktacja za część zadania	Maksymalna punktacja za zadanie
1	a	130000_8	1	1	5
	b	3A3B3C	1	1	
	c	Pascal	1	1	
	d	Sortowanie przez scalanie (Merge Sort)	1	1	
	e	3, 1, 4, 2	1	1	

Zadanie 2 — Liczby Fibonacciego (6 pkt.)

<table>
<tr><th>Numer zadania</th><th>Część zadania</th><th>Czynność</th><th>Maksymalna punktacja za czynność</th><th>Maksymalna punktacja za część zadania</th><th>Maksymalna punktacja za zadanie</th></tr>
<tr><td rowspan="3">2</td><td>a</td><td>Przykładowy program. Punkt za użycie funkcji rekurencyjnej w programie, inaczej brak punktu. (Pascal)
<pre>Program Fibonacci_rek;
 uses crt;
 var n:integer;
 function
Fib_rek(n:integer):integer;
  begin
   if(n = 1) or (n = 2) then
Fib_rek:=1
   else Fib_rek:= Fib_rek(n-1) +
Fib_rek(n-2)
  end;
begin
 read(n);
 write(n,' ',Fib_rek(n))
end.</pre></td><td>1</td><td>1</td><td rowspan="3">6</td></tr>
<tr><td rowspan="2">b</td><td>Przykładowy program.
<pre>Program Fibonacci_iter;
 uses crt;
 var n:integer;
 function
Fib_iter(n:integer):integer;
  var f1,f2,i:integer;
  begin
   if(n = 1) or (n = 2) then
Fib_iter:=1
   else begin
    f1:=1; f2:=1;
    for i:=3 to n do begin
     f2:=f1+f2;
     f1:=f2-f1
    end;
    Fib_iter:=f2
   end
  end;
begin
 read(n);
 write(n,' ',Fib_iter(n))
end.</pre></td><td>2</td><td rowspan="2">2</td></tr>
<tr><td>Tylko jeden punkt, jeśli błąd w operowaniu wartościami poprzednich liczb Fibonacciego.</td><td>1</td></tr>
</table>

	c	Trzy razy w przypadku rekurencji i raz w przypadku iteracji. Ta pierwsza konkluzja powinna być wyprowadzona z drzewa wywołań rekurencyjnych.	2	2	
		Podanie poprawnej liczby tylko w jednym z przypadków.	1		
	d	Poprawna interpretacja powinna uwzględniać drzewo wywołań funkcji rekurencyjnej i to, że te wywołania są realizowane niezależnie jedno od drugiego. Uczeń powinien podać co najmniej jedną z konsekwencji takiej sytuacji: algorytm rekurencyjny działa wolniej i zużywa więcej pamięci.	1	1	

Zadanie 3 — Progi i schody (9 pkt.)

<table>
<tr><th>Numer zadania</th><th>Część zadania</th><th>Czynność</th><th>Maksymalna punktacja za czynność</th><th>Maksymalna punktacja za część zadania</th><th>Maksymalna punktacja za zadanie</th></tr>
<tr><td rowspan="9">3</td><td>a</td><td>Poprawne wartości w kolejności:
schody do dołu, ich długość, liczba progów
2, 2, 2 3 0
3, 1, 1 3 1
3, 3, 1 3 1
11,7, 7, 6 4 2
7, 7 2 0
9, 9, 7 3 1</td><td>1</td><td>1</td><td rowspan="9">9</td></tr>
<tr><td rowspan="2">b</td><td>Przykładowy program; program nie musi zawierać tablicy, by przechowywać dane, można zliczać schody czytając elementy bez ich przechowywania.

(Pascal)
<pre>program Progi;
 uses crt;
 var i,j,n:integer;
     a:array[1..100] of integer;
begin
 readln(n);
 for i:=1 to n do readln(a[i]);
 j:=0;
 for i:=1 to n-1 do
  if a[i] > a[i+1] then j:=j+1;
 write('Ciag ma ',j,' schodow')
end.)</pre></td><td>2</td><td rowspan="2">2</td></tr>
<tr><td>Program, który uwzględnia nie wszystkie elementy ciagu, a w konsekwencji nie znajduje wszystkich progów w ciągu.</td><td>1</td></tr>
<tr><td rowspan="5">c</td><td>Prawidłowe wartości początkowe dla zmiennych</td><td>1</td><td rowspan="5">5</td></tr>
<tr><td>Uwzględniono cały zakres danych</td><td>1</td></tr>
<tr><td>Prawidłowa nierówność dla schodów do dołu</td><td>1</td></tr>
<tr><td>Prawidłowe zliczenie liczby schodów w podciągach</td><td>1</td></tr>
<tr><td>Prawidłowe znalezienie największej liczby progów</td><td>1</td></tr>
</table>

<table>
<tr><td></td><td></td><td>Przykładowe rozwiązania:

(C/C++)

(Pascal)
<pre>
program Progi_w_schodach;
 uses crt;
 var i,j,k,n:integer;
     a:array[1..100] of integer;
begin
 readln(n);
 for i:=1 to n do readln(a[i]);
 j:=0;
 k:=0;
 for i:=1 to n-1 do
  if a[i] >= a[i+1] then
   if a[i] > a[i+1] then j:=j+1
  else
   if j > k then begin
    k:=j;
    j:=0
   end;
   if j > k then k:=j;
  write('Najwieksza liczba progow w
schodach do dolu = ',k)
end.
</pre></td><td></td><td></td></tr>
<tr><td></td><td>d</td><td>Poprawny algorytm w punkcie c) powinien sprawdzać, czy kolejne elementy tworzą schody (nierówność słaba między elementami ≤) a następnie, czy tworzą próg (nierówność <). Razem, w najlepszym przypadku, algorytm może wykonywać $2(n-1)$ porównań. Liczba porównań powinna być liniowa względem n. Punkt otrzymuje rozwiązanie, w którym uczeń objaśnia, ile porównań wykonuje jego algorytm.</td><td>1</td><td>1</td><td></td></tr>
</table>

Organizatorzy:
Polskie Towarzystwo Informatyczne Oddział Kujawsko-Pomorski
Uniwersytet Mikołaja Kopernika w Toruniu Wydział Matematyki i Informatyki
Centrum Kształcenia Ustawicznego TODMiDN w Toruniu

SCHEMAT OCENIANIA
poziom rozszerzony
arkusz II

Zadanie 4 – Pogotowie ratunkowe (10 pkt)

Numer zadania	Część zadania	Czynność	Maksymalna punktacja za czynność	Maksymalna punktacja za część zadania	Maksymalna punktacja za zadanie
4	a	Sporządzenie listy realizowanych usług medycznych i obliczenie liczby każdej z nich .	2	4	10
		Wskazanie lekarza, który najczęściej udzielił pomocy .	2		
	b	Wyznaczenie numeru miesiąca na podstawie daty przyjęcia usługi medycznej	1	3	
		Wyznaczenie kosztu usługi medycznej w każdym miesiącu dla kryterium refundacji przez NFZ	2		
	c	Obliczenie kosztu pomocy poniesionej przez każdego ubezpieczyciela	1	3	
		Sporządzenie czytelnego wykresu	2		

Przykładowe rozwiązanie znajduje się w pliku ***pogotowie_roz.xls.***

Zadanie 5 — Biblioteka (10 pkt)

Numer zadania	Część zadania	Czynność	Maksymalna punktacja za czynność	Maksymalna punktacja za część zadania	Maksymalna punktacja za zadanie
5	a	Podanie średniego wieku książki 64,405 lat, udokumentowane załączoną stosowną realizacją komputerową	2	2	10
	b	Udokumentowane załączoną realizacją komputerową poprawne i zgodne ze specyfikacją zadania (patrz przykładowe rozwiązanie) utworzenie list	1 pkt. za każdą listę	6	
	c1	Podanie poprawnej i zgodnej ze specyfikacją zadania (patrz przykładowe rozwiązanie) listy Uwaga! Ostatnia osoba z listy ma 9 wypożyczeń – jest 9 osób mających po 9 wypożyczeń,. Pytanie nie zawiera dodatkowych kryteriów, a więc na liście może pojawić się każda z owych 9 osób	1	1	
	c2	Podanie poprawnej i zgodnej ze specyfikacją zadania (patrz przykładowe rozwiązanie) listy Uwaga! Ostatnie 4 książki z listy mają 9 wypożyczeń – jest 10 książek mających po 9 wypożyczeń,. Pytanie nie zawiera dodatkowych kryteriów, a więc na liściemoże pojawić się dowolny zestaw 4 z owych 10 książek mających po 9 wypożyczeń		1	

Uwaga! Treść zadania nie zawiera dyspozycji co do wyboru narzędzia. Zadanie może być rozwiązane przy pomocy różnych narzędzi informatycznych: bazy danych, arkusza kalkulacyjnego.

Przykładowe rozwiązanie znajduje się w pliku ***biblioteka_roz.mdb***.

Zadanie 6 Anagramy cyfrowe (10 pkt)

Numer zadania	Część zadania	Czynność	Maksymalna punktacja za czynność	Maksymalna punktacja za część zadania	Maksymalna punktacja za zadanie
6	a	Podanie prawidłowych liczb	1	1	10
	b	Prawidłowe wyodrębnienie cyfr z liczb	1	5	
		Zaprogramowanie zliczania lub sortowania cyfr w liczbach	1		
		Zastosowanie funkcji zliczającej lub sortującej	1		
		Porównanie wyników zliczenia lub posortowanych cyfr	1		
		Prawidłowa odpowiedź w zależności od wyniku porównania	1		
	C	Prawidłowe nazwy plików, do których odwołuje się program	1	4	
		Prawidłowe czytanie z pliku	1		
		Prawidłowe obliczenia	1		
		Prawidłowe zapisanie wyników do pliku	1		

Przykładowa realizacja algorytmu z punktu b):

```
#include <iostream>
using namespace std;
void zlicz(int a, int *t)
{
   while (a>0)
   {
      t[a%10]++;
      a=a/10;
   }
}
bool anag(int *t, int *p)
{
   bool jest=true, nie_jest=false;
   int i=0;
   for (i=0; i<10; i++)
   if (t[i]!=p[i]) return nie_jest;
   return jest;
}
int main()
{
   int x, y;
   cout<<"Podaj pierwsza liczbe";    cin >>x;
   cout<<"Podaj druga liczbe";    cin>>y;
```

```
    int tx[10], ty[10];
    for (int i=0; i<10; i++) tx[i]=0;
    for (int i=0; i<10; i++) ty[i]=0;
    zlicz(x, tx);
    zlicz(y, ty);
    if (anag(tx,ty)) cout<<"Te liczby sa anagramami";
    else cout<<"Te liczby nie sa anagramami";
    return 0;
}
```

Przykładowa realizacja algorytmu z punktu c) (również w pliku anagramy_pliki.cpp):

```
#include <iostream>
#include <fstream>
using namespace std;
void zlicz(int a, int *t)
{
    while (a>0)
    {
        t[a%10]++;
        a=a/10;
    }
}
bool anag(int *t, int *p)
{
    bool jest=true, nie_jest=false;
    int i=0;
    for (i=0; i<10; i++)
    if (t[i]!=p[i]) return nie_jest;
    return jest;
}
int main()
{
    int n, x, y;
    int tx[10], ty[10];
    ifstream p("dane.txt");
    ofstream q("anagramy.txt");
    p >>n;
    for (int i=1; i<=n; i++)
    {
        p>>x;
        p>>y;

        for (int i=0; i<10; i++) tx[i]=0;
        for (int i=0; i<10; i++) ty[i]=0;
        zlicz(x, tx);
        zlicz(y, ty);
       q<<x<<" "<<y<<" ";
        if (anag(tx,ty)) q<<"tak"<<endl; else q<<"nie"<<endl;
    }
    p.close();
    q.close();
    return 0;
}
```