1. Rozwiązać kryptograf (*):

$$\begin{array}{cccccc} & & S & E & N & D \\ + & & M & O & R & E \\ \hline & M & O & N & E & Y \end{array}$$

2. Napisać program sortujący listy przez tworzenie drzewa z elementów listy. Kolejno pobierane elementy listy porównywane są z wierzchołkami tworzonego drzewa i przesuwane do lewych lub prawych gałezi w zależnosci od tego czy sa mniejsze czy wieksze od istniejacych już wierzchołkow. Po utworzeniu drzewa następuje wypisanie jego wierzchołków w postaci listy. (**)
3. Wyznaczyć w grafie wszystkie spójne podgrafy (w postaci listy list wierzchołków), tzn. takie w których pomiędzy dwoma dowolnymi ich wierzchołkami istnieje droga. (*)
4. Wyznaczyć w grafie wszystkie podzbiory wierzchołków, w których każdy wierzchołek jest połączony z każdym. (*)
5. Wyznaczyć w grafie zorientowanym wszystkie jego cykle (w postaci listy list wierzchołków). (*)
6. Wykreślić zadaną figurę bez odrywania ręki (poprzez podanie listy wierzchołków). (*)
7. Obliczyć wartość wyrażenia logicznego zadanego w postaci łańcucha operatorów and, or, not i nawiasów. (**)
8. Przygotować pakiet programów dotyczących operacji na drzewach. Pakiet powinien zawierać predykaty: określające przynależność wierzchołka do drzewa, tworzące listę wierzchołków o różnych kolejnościach, dodające drzewa, wycinające z drzewa gałąź rozpoczynającą się od wskazanego wierzchołka, wyznaczające liczbę wierzchołków, zamieniające drzewa na pary wierzchołków, zamieniające pary wierzchołków na drzewo. (**)
9. Program znajdujący metodą sita Eratostenesa listę liczb pierwszych spośród liczb naturalnych z przedziału $[1, N]$. (*)
10. Program znajdujący rozwiązania równania Pitagorasa dla liczb całkowitych. (*)
11. Program znajdujący rozwiązania gry „3×3” metodą szukania w głąb. (*)
12. Przygotować program wyznaczający drogę konika szachowego obchodzącego wszystkie pola szachownicy o rozmiarach $N \times N$ i przebywającego na każdym polu tylko jeden raz. (*)
13. Rozwiązać zadanie o hetmanach (N-queens problem) – umieścić N hetmanów na szachownicy o rozmiarach $N \times N$ tak, aby nawzajem się nie atakowały. (*)
14. Wyznaczyć optymalny ruch w grze NIM, która polega na kolejnym pobieraniu przez przeciwników przynajmniej jednej z zapałek z jednego spośród kilku stosów. Grę przegrywa gracz, który nie może wykonać ruchu. (*)
15. Istnieją dwie liczby M i N, takie że $1 < M < N < 100$. Panu S powiedziano sumę tych liczb, panu I iloczyn tych liczb i obaj wiedzieli, że im te wartości podano. Wywiązał się między nimi następujący dialog:

    I: Nie znam tych liczb.

    S: Wiedziałem, że ich nie znasz. Ja nie znam ich także.

    I: Teraz już wiem jakie to są liczby.

    S: Teraz ja także wiem.

    Przygotować program znajdujący te liczby. (***)
16. Wyznaczyć optymalny ruch w grze w kółko i krzyżyk na planszy 3x3. (*)
17. Program grający w gomoku. (**)
18. Program grający w renju. (**)
19. Program sortujący elementy listy metodą pęcherzykową (bubble sort).
20. Na podstawie listy symboli utworzyć listę liczb całkowitych określającą pozycję poszczególnych symboli w hierarchii starszeństwa. Np. liście `[p,r,o,l,o,g]` odpowiada lista `[4,5,3,2,3,1]`. (*)
21. Program znajdujący najkrótszą drogę między dwoma miastami. (**)

22. Poniżej podane sa imiona trzech muzyków, ich nazwiska, instrumenty na których grają, okres czasu przez który grają oraz cztery dotyczące ich fakty:

imiona	adam	michał	ryszard
nazwiska	cicho	powoli	szybko
instrumenty	flet	puzon	skrzypce
czas	trzy	cztery	pięć

    1. ryszard gra dłużej od osoby grającej na flecie.
    2. michał powoli nie gra cztery lata.
    3. pan cicho gra pięć lat ale nie na puzonie.
    4. adam nie nazywa się szybko.

    Napisać program wiążący imiona muzyków z nazwiskami, instrumentami i czasem gry. (*)

23. Przygotować program obliczający wartość wyrażeń arytmetycznych składających się z operacji „+", „−", „×" i „/" zadawanych w postaci listy lub łańcucha. (*)

24. Program sprawdzający czy zadawane zdania należą do gramatyki angielskiej:

```
sentence --> noun_phrase,verb_phrase,complement.
```

25. Program przestawiający bloki. Bloki mogą posiadać długość 1 lub 2 i ułożone są wdłuż linii podzielonej na odcinki o długości 1. Bloki mogą być kładzione na innych, przy czym bloki o długości 2 mogą być kładzione wyłącznie na dwóch przylegających blokach. Program winien realizować komendy:

```
poloz_blok_na_miejsce(B,M),
zwolnij_miejsce(M),
poloz_blok_na_blok(B1,B2).
```

    Położenie bloków przedstawiane jest graficznie. (**)

26. Program obliczający wyniki operacji dodawania, odejmowania, mnożenia i dzielenia ułamków. (*)

27. Program tworzący lustrzane odbicie zadanego drzewa. (*)

28. Program realizujący rotację listy w zadanym kierunku o zadaną liczbę miejsc. (*)

29. Program wydający resztę z zapłaconej kwoty w możliwie małej liczbie banknotów i monet. Przyjąć istnienie nominałów: 200, 100, 50, 20, 10, 5, 2, 1, 0.5, 0.2, 0.1, 0.05, 0.02 i 0.01. (*)

30. Program obliczający ilość dni pomiędzy zadanymi datami. Program powinien sprawdzać także poprawność dat. (*)

31. Program wczytujący co najwyżej czterocyfrowe liczby całkowite i podający wartość tych liczb w językach polskim i angielskim. (*)

32. Program zamieniający liczby rzymskie na arabskie i odwrotnie. (*)

33. Program zamieniający liczby binarne (zadane w postaci łańcuchów) na liczby w kodzie Gray'a i odwrotnie. (*)

34. Program generujący wszystkie słowa kodu dwójkowego o stałej liczbie jedynek (kodu k z n). (*)

35. Przygotować pakiet programów dotyczących operacji na drzewach. Pakiet powinien zawierać predykaty: określające przynależność wierzchołka do drzewa, tworzące listę wierzchołków o różnych kolejnościach, dodające drzewa, wycinajace z drzewa galaz rozpoczynajaca sie od wskazanego wierzcholka, wyznaczające liczbę wierzchołków, zamieniające drzewa na pary wierzchołków, zamieniające pary wierzchołków na drzewo. (**)

36. Program znajdujący metodą sita Eratostenesa listę liczb pierwszych spośród liczb naturalnych z przedziału $[1, N]$. (*)

37. Program znajdujący rozwiązania gry „3×3" metodą szukania w głąb. (*)

38. Przygotować program wyznaczający drogę konika szachowego obchodzącego wszystkie pola szachownicy o rozmiarach $N \times N$ i przebywającego na każdym polu tylko jeden raz. (*)

39. Rozwiązać zadanie o hetmanach (N-queens problem) – umieścić N hetmanów na szachownicy o rozmiarach $N \times N$ tak, aby nawzajem się nie atakowały. (*)

40. Istnieją dwie liczby M i N, takie że $1 < M < N < 100$. Panu S powiedziano sumę tych liczb, panu I iloczyn tych liczb i obaj wiedzieli, że im te wartości podano. Wywiązał się między nimi następujący dialog:

    I: Nie znam tych liczb.

    S: Wiedziałem, że ich nie znasz. Ja nie znam ich także.

    I: Teraz już wiem jakie to są liczby.

    S: Teraz ja także wiem.

    Przygotować program znajdujący te liczby. (***)

41. Wyznaczyć optymalny ruch w grze w kółko i krzyżyk. (*)

42. Program sortujący elementy listy metodą pęcherzykową (bubble sort). (*)

43. Na podstawie listy symboli utworzyć listę liczb całkowitych określającą pozycję poszczególnych symboli w hierarchii starszeństwa. Np. liście `[p,r,o,l,o,g]` odpowiada lista `[4,5,3,2,3,1]`. (*)

44. Program symulujący działanie sieci zbudowanej z bramek NAND. Sieć jest zadana grafem skierowanym reprezentującym połączenia pomiędzy bramkami. (*)

45. Program kolorujący przy pomocy czterech barw przylegające do siebie obszary w przestrzeni dwuwymiarowej. Przyleganie obszarów do siebie oraz rezultat kolorowania zadawane są listami par. (*)

46. Program minimalizujący funkcje logiczne metodą Quine'a-McCluskey'a. (***)

47. Program znajdujący wszystkie 5 cyfrowe liczby, których suma cyfr wynosi 21. (*)

48. Program tasujący i rozdający talię kart czterem graczom. (*)

49. Wyznaczyć wszystkie kwadraty magiczne o wymiarach $N \times N$. (*)

50. Przedstawiajac macierz jako listę list reprezentujacych wiersze zrealizowac dodawanie i transpozycje macierzy. (*)

51. Przedstawiajac macierz jako listę list reprezentujacych wiersze zrealizowac mnożenie dwóch macierzy. (*)

52. Program porównujący siłę kart na dwóch rękach przy grze w pokera. Program sprawdza także legalność kart. (*)

53. Program kolorujący węzły grafu. (*)

54. Program rozwiązujący zadanie o komiwojażerze. (*)

55. Program rozkładający liczbę całkowitą na czynniki oraz wyznaczający największy wspólny podzielnik i najmniejszą wspólną wielokrotność pary liczb. (*)

56. Program podający wzór na obliczenie wyznacznika macierzy potrafiący zerować odpowiednie iloczyny w przypadkach, gdy niektóre elementy macierzy równe są zeru. (**)

57. Program sortujacy listę usprawnioną metodą pęcherzykową. (*)

58. Program rozwiązujący równania liniowe z jedną niewiadomą o współczynnikach będących liczbami całkowitymi. Program nie powinien dokonywać obliczeń na ułamkach dziesiętnych. (*)

59. Program określający czy usunięcie w grafie zadanego łuku łączącego dwa wierzchołki spowoduje rozpad grafu na dwa odrębne grafy. (*)

60. Program różniczkujący wyrażenia zawierajace operacje „+”, „−”, „×”, „/”, oraz funkcje log, exp i trygonometryczne. (**)

61. Dane są dwa naczynia, siedmoilitrowe i pięciolitrowe, oba poczatkowo puste. Należy wyznaczyć ciąg akcji napełniania, opróżniania i przelewania, po których w większym naczyniu znajdą się 4 litry wody. (*)

62. Przygotować program generujący losowo wyrażenia arytmetyczne zawierające liczby całkowite, nawiasy oraz operację „+”, „−”, „×”, „/”. (*)

63. Program obliczający rezystancję pomiędzy dwoma zadanymi elementami sieci rezystorów. (*)

64. Parser sprawdzający poprawność wyrażen arytmetycznych zawierających liczby, symbole, nawiasy oraz operacje arytmetyczne. (*)
65. Parser języka Pascal. (**)
66. Parser języka C. (**)
67. Program przekształcający dowolne wyrażenia predykatowe (wpz) do postaci klauzul. Wyrażenia wejściowe i wyjściowe zadawane są w postaci łańcuchów. (**)
68. Program dowodzący wynikanie logiczne metodą rezolucji z dowolnie przyjętą strategią szukania. (**)
69. Program obliczający pola powierzchni figur na podstawie zadawanych niektórych ich rozmiarów i kątów. (*)
70. Program wyznaczający najlepszy ruch w grze REVERSI na podstawie analizy jej pełnego grafu. (**)
71. Program wyznaczający najlepszy ruch w dowolnej grze stosując strategię minimax o zadanej głębokości. (**)
72. Program rysujący na ekranie drzewo binarne zadane jako obiekt o strukturze złożonej. (*)
73. Program korygujący tekst (string) przez eliminację krotności występowania wyrazu w przypadku gdy powtarza się on kilka razy z rzędu. (*)
74. Program obliczający wartość wyrażeń arytmetycznych składających się z liczb całkowitych, nawiasów, oraz operacji arytmetycznych „+", „−", „×", „/". Program przeprowadza obliczenia nie korzystając z ułamków dziesiętnych. (**).
75. Program przeprowadzający redukcję wyrazów podobnych i wyłączanie przed nawias czynników wspólnych dla wszystkich wyrazów w wielomianach zadanych w postaci sumy wyrażeń. (*)
76. Program przeprowadzający mnożenia wielomianów zadanych w postaci sumy wyrażeń i redukcję wyrazów podobnych w wynikach. (*)
77. Program znajdujący rozwiązania gry „3×3" metodą szukania uporządkowanego (heurystycznego). Jako kryterium przyjąć funkcję $h(n) = P(n) + 3S(n) + 1$, gdzie $P(n)$ jest sumą odległości klocków od swoich końcowych pozycji a $S(n)$ jest obliczane przez obiegnięcie wszystkich niecentralnych klocków i przyznawanie liczby 0 gdy klocek następuje po właściwym klocku i liczby 2 w przeciwnym przypadku. (**)
78. Program stosujący wzory redukcyjne sprowadzający funkcję trygonometryczne dowolnego kąta do funkcji kąta w pierwszej ćwiartce. (*)
79. Program sprawdzający tożsamości trygonometryczne. (**)
80. Wyznaczyć krotność zadanego elementu listy.
81. Zdefiniować predykat odczytujący z listy element o zadanej pozycji lub pozycję zadanego elementu.
82. Rozłożyć listę na element o wskazanej pozycji i listę pozostałych elementów.
83. Wstawić element na wskazaną pozycję w liście bez usuwania elementu będącego na tej pozycji.
84. Wstawić zadaną listę na wskazaną pozycje w liście.
85. Zamienić: a) element na wskazanej pozycji w liście na element zadany, b) dwa elementy listy na wskazanych pozycjach pomiędzy sobą.
86. Sprawdzić, czy zadany element występuje przynajmniej raz na parzystej pozycji w liście.
87. Wyznaczyć maksymalny element listy.
88. Rozłożyć listę na element minimalny i listę pozostałych elementów (mogą być w zmienionej kolejności).
89. Podzielić listę na dwie listy: a) pierwsza zawiera zadaną liczbę elementów, b) pierwsza kończy się zadanym elementem.
90. Podzielić listę na wszystkie możliwe podlisty. Wynik przedstawic w postaci listy podlist. (*)
91. Wygenerować wszystkie k-elementowe kombinacje elementów zadanej listy.
92. Podzielić elementy listy na wszystkie możliwe podzbiory. (*)
93. Wyznaczyć wszystkie k-elementowe wariacje elementów listy, gdzie k jest zadaną liczbą.

94. Wyznaczyć iloczyn logiczny dwóch list.

95. Wyznaczyć sumę logiczną dwóch list.

96. Zdefiniować operacje algebry Boole'a: `not`, `and`, `or`, `nand`, `nor`, `xor`, `nxor`, `implikacja`, dla podzbiorów zbioru 12 liter alfabetu. (*)

97. Korzystając z generatora liczb losowych ustawić w losowej kolejności (przetasować) elementy listy.

98. Wyznaczyć odległość dwóch list o tej samej długości jako liczbę występujących na tych samych pozycjach różniących się między sobą elementów.

99. Przygotować program symulujący działanie bramek logicznych NOT, AND, OR, NAND, NOR, XOR o dowolnej liczbie wejść (pierwszym argumentem odpowiednich predykatów jest lista reprezentująca wejścia, zaś drugim argumentem, wyjście). Korzystając ze zdefiniowanych bramek zbudować sumator i układ sprawdzający, czy czterobitowa liczba wejściowa może być kodem BCD (tzn. czy jest zawarta w przedziale od 0 do 9). (*)

100. Przygotować program wykreślający na ekranie fraktal „krzywa smoka" (obrazujący krawędź paska papieru złożonego N krotnie w pół, a następnie rozłożonego tak, aby zgięcia papieru tworzyły kąty proste). Kreśli się go przesuwając na ekranie pisak (żółwia) o stałą odległość w kierunkach zadanych kolejnymi elementami listy. Listę tę, `(L)`, której elementami są kierunki e,n,w,s (wschód, północ, zachód, południe), konstruuje się w procesie rekurencyjnym o zadanej liczbie kroków, `(N)`, w następujący sposób: W kroku `N=0`, `L=[e]`. W następnych krokach każdy element listy zastąpiony zostaje dwoma elementami wg. następujących reguł produkcji: (**)

```
e --> en,  n --> wn,  w --> ws,  s --> es
```

101. Przygotować program konstruujący pełne drzewo binarne o zadanej głębokości z węzłami ponumerowanymi kolejnymi liczbami naturalnymi. (*)

102. Przygotować program kreślący na ekranie zadane drzewo bez wyświetlania numerów wzłów; np. drzewo uzyskane w poprzednim zadaniu. (*)

103. Zdefiniować predykat `drzewo(D)` konstruujący drzewo binarne `D` na podstawie wczytywanych z klawiatury kolejnych numerów jego wzłów, przy czym wciśnięcie klawisza `Esc` sygnalizuje brak potomnego wierzchołka generowanego przez ostatnio wprowadzony wierzchołek. Przyjąć następujące deklaracje: (*)

```
domains
  galaz=g(integer,galaz,galaz); n
predicates
  drzewo(galaz)
```

104. Przygotować program rysujący powstające drzewo binarne w trakcie wczytywania jego kolejnych węzłów. Przyjąć, że węzły drzewa nie mają oznaczeń, i że wprowadzane są naciskaniem dowolnych klawiszy, przy czym naciśnięcie klawisza `n` oznacza brak potomnego wierzchołka. (**)

105. Przygotować program tworzący sumę dwóch drzew binarnych. Program dodaje do drzewa drugie drzewo we wskazanym miejscu, którym jest nazwa węzła. Jeżeli z wskazanego węzła generowane są już dwa węzły potomne, to program nie wykonuje się. Przyjąć deklaracje: (*)

```
domains
  galaz=g(integer,galaz,galaz); n
predicates
  drzewo(galaz)
  suma(integer,galaz,galaz,galaz)
```

106. Zdefiniować predykat drzewo konstruujący drzewo binarne na podstawie, zadanych w predykatach `para`, uporządkowanych par węzłów. Zakłada się, że jeden z węzłów nie posiada węzła rodzicielskiego (jest korzeniem) i program powinien go odszukać, a następnie skonstruować drzewo. Przyjąć następujące deklaracje: (*)

```
domains
  galaz=reference g(symbol,galaz,galaz); n
predicates
  drzewo(galaz)
  para(symbol,symbol)
```

107. Przygotować program sprawdzający istnienie wskazanego pokrewieństwa pomiędzy zadanymi osobami (ojciec, matka, brat, siostra, rodzeństwo, ciotka, wuj, kuzyn, kuzynka, babcia, dziadek, wnuk, wnuczka, przodek) oraz podający pokrewienstwo pomiędzy zadanymi osobami. Program zawiera bazę danych w postaci zbioru predykatów `rodzice(ojciec,matka,dziecko,plec)` oraz reguły definiujące pokrewieństwo.
108. Przygotować program generujący kolejne posunięcia w układance Wieże Hanoi. (*)
109. Napisać program rozwiązujący zadanie o misjonarzach i ludożercach: na lewym brzegu rzeki znajduje się N misjonarzy i N ludożerców oraz łódź mieszcząca dwie osoby. Należy zaplanować bezpieczną przeprawę przez rzekę unikając sytuacji, w których na którymkolwiek brzegu byłoby więcej ludożerców niż misjonarzy. (**)
110. System „świat smoka": przygotować program wnioskujący o występowaniu smoka w zadanym kwadracie na podstawie zadanych faktów dotyczących istnienia, nieistnienia i braku informacji o zapachu w kratkach systemu. (**)
111. Napisać program sprawdzający czy lista jest podlistą listy.
112. Przygotować program zamieniający w liście zadaną podlistę przez inną listę. (*)
113. Napisać program w prologu obliczający pole dowolnej figury płaskiej na podstawie współrzędnych jej wierzchołków. (*)
114. Napisać w prologu program zamieniający zadaną kwotę na jej reprezentację słowną. (*)
115. Napisać w prologu program (podobny do grep'a), który będzie w zadanym tekście wyszukiwał wiersze zawierające zadany wzorzec. Wzorzec może zawierać symbole wieloznaczne. (*)
116. Napisać w prologu program znajdujący drogę konika szachowego między zadanymi polami szachownicy. (*)
117. Napisać w prologu program zamieniający liczby arabskie na liczby rzymskie. (*)
118. Wyznaczanie wolnych pól w grze Miner. (**)
119. Rozwiązywanie pasjansa FreeCell. (**)
120. Program grający w szachy. (**)
121. Rozwiązywanie zagadek szachowych typu „mat w 2 ruchach". (**)
122. Program grający w warcaby polskie. (**)
123. Układanie krzyżówek na podstawie zadanego słownika i rozmiarów planszy. (**)
124. Program ustalający pierwszeństwo przejazdu na dowolnych skrzyżowaniach i w dowolnych sytuacjach zgodnie z obowiązującymi w Polsce zasadami ruchu drogowego. (**)
125. Implementacja automatu komórkowego LIFE J. Conwaya z wykorzystaniem grafiki. (*)
126. Program do szukania pierwiastków całkowitych wielomianów.
127. Program, który dla zadanego N wyznacza listę/y N liczb L_i, $i = 0 \ldots N-1$, takich że $L_i = \sum_{k=0}^{N-1} L_k == i$, czyli każdy wyraz ciągu odpowiada liczbie wystąpień elementów równych indeksowi wyrazu. Np. dla $N = 4$: $L = [1, 2, 1, 0]$ lub $L = [2, 0, 2, 0]$ dla $N = 5$: $L = [2, 1, 2, 0, 0]$, dla $N = 6$: brak rozwiazań.
128. Program ustalający nazwę miasta po jak najmniejszej liczbie pytań z odpowiedziami tak lub nie (co najmniej 40 miast). Np. czy przez miasto przepływa rzeka?