Imię i nazwisko: ..... **WdI, Kolokwium nr 2**
Indeks: .....

1. **[7]** Dana jest następująca deklaracja:

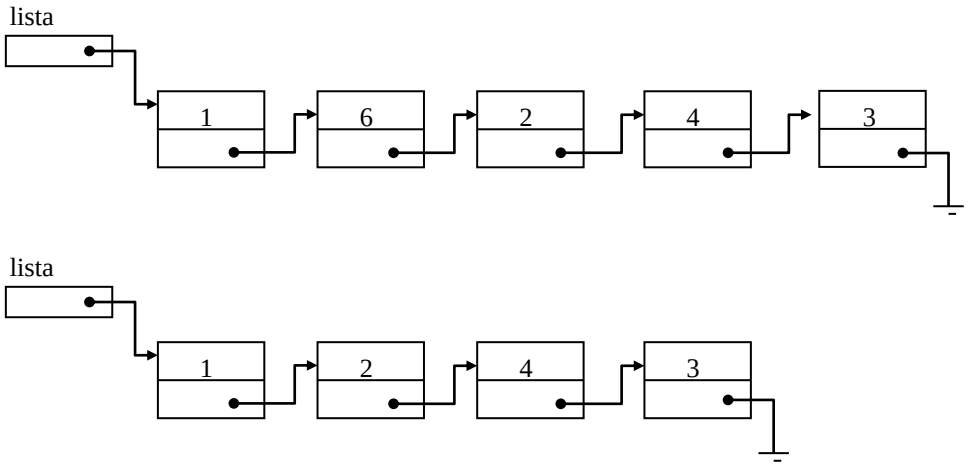
```
struct elem { int val; struct elem *next; };
```

```
class ListItem:
 def __init__(self,value): self.val = value; self.next = None
```

Wersja 1 [7]: Napisz funkcję `usunMax`, która z listy podanej jako argument usunie element o największej wartości `val` i zwróci wynikową listę jako wartość. Np. dla listy składającej się z elementów `1, 6, 2, 4, 3` wywołanie `usunMax(lista)` powinno dać w wyniku listę o elementach `1, 2, 4, 3`. Nagłówek funkcji w języku C powinien mieć postać

```
struct elem *usunMax(struct elem *lista)
```

a w Pythonie `def usunMax(lista)`. Możesz przyjąć, że wartości elementów na liście nie powtarzają się.



Rys. Górny rysunek przedstawia sytuację przed wykonaniem funkcji, a dolny – po wykonaniu.

Wersja 2 [4]: Napisz funkcję `gdzieMax`, która dla listy podanej jako argument zwraca 1 gdy element o największej wartości val jest pierwszym lub ostatnim elementem listy, a 0 w przeciwnym przypadku. Np. dla listy 1, 6, 2, 4, 3 z rysunku wartość powinna wynieść 0, a dla list 10, 2, 4, 7 oraz 10, 9, 8, 40 wartość to 1.

Imię i nazwisko: ..... **WdI, Kolokwium nr 2**
Indeks: .....

2. **[7]** Drzewo binarne nazywać będziemy *kopcowym*, jeśli spełnia ono następujące warunki:
   a. Każdy węzeł ma dwójkę dzieci (lewe i prawe) lub w ogóle nie ma dzieci.
   b. Dla każdego węzła, który ma dzieci, wartości val w dzieciach są większe niż wartość val w danym węźle.

   Na przykład drzewa A i C z poniższego rysunku są kopcowe, natomiast drzewo B nie jest kopcowe (węzeł o wartości 8 ma tylko jedno dziecko), drzewo D również nie jest kopcowe (wartość w lewym dziecku węzła 5 wynosi 3<5).

   Korzystamy z następujących deklaracji:

   - Język C:

```c
typedef struct node *pnode;
typedef struct node{
        int val;
        pnode left;
        pnode right;} snode;
```

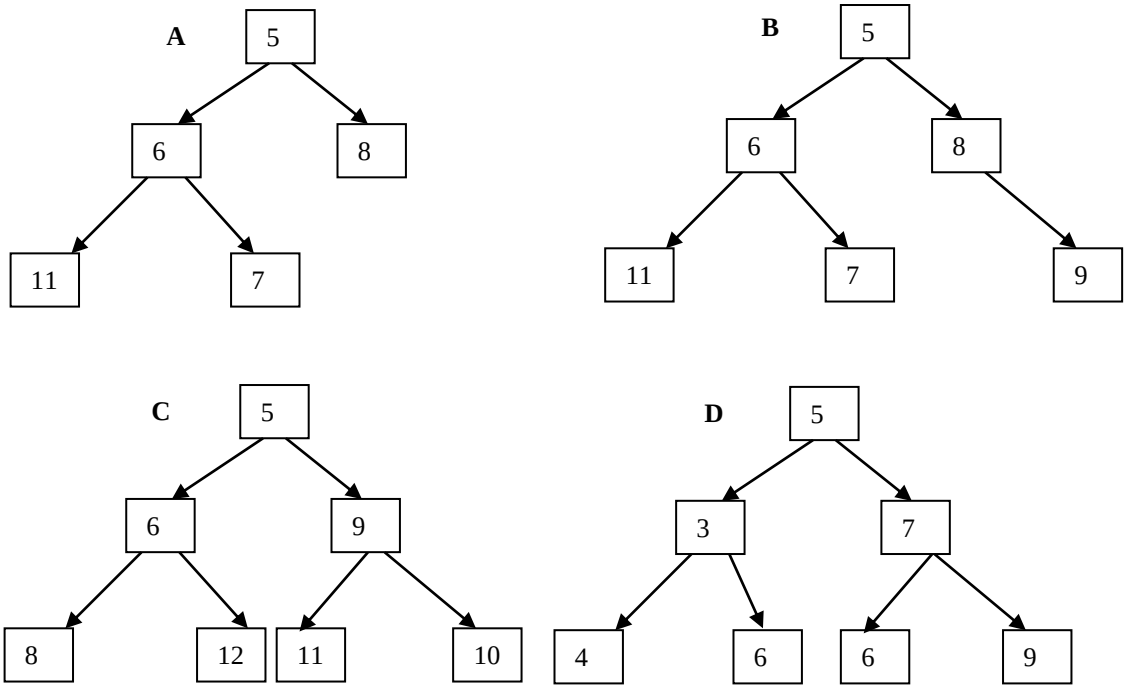
   - Język Python:

```python
class TreeItem:
   def __init__(self,value):
      self.val = value
      self.left = None
      self.right = None
```

   Napisz funkcję `czyKopiec`, która dla drzewa podanego jako argument zwróci wartość true (lub 1) gdy drzewo jest kopcowe oraz fałsz (lub 0) w przeciwnym wypadku.
   Nagłówek funkcji w języku C powinie mieć postać

```c
bool czyKopiec(pnode drzewo)
```

   a w Pythonie `def czyKopiec(drzewo)`



Imię i nazwisko: ..... **WdI, Kolokwium nr 2**
Indeks: .....

3. **[5]** Dane są następujące funkcje:

```
(01) bool pr(int i, int j, int a[])
(02) { if (i==j) return true;
(03)   s = (i+j) / 2;
(04)   if (a[s]>a[s+1) return false;
(05)   bool p1 = pr(i,s,a)
(06)   bool p2 = pr(s+1,j,a);
(07)   return p1 && p2;
(08) }
(09
(10) int prt(int n, int a[]){
(11)   return pr(0,n-1,a);
(12) }
```

```
def pr(i, j, a):
  if i==j: return true
  s = (i+j) / 2
  if (a[s]>a[s+1):
    return false
  p1 = pr(i,s,a)
  p2 = pr(s+1,j,a)
  return p1 && p2

def prt(n, a):
  return pr(0,n-1,a)
```

a) [1] Podaj drzewo wywołań rekurencyjnych dla `pr(9,18,a)`

b) [2] Uzupełnij specyfikację funkcji `prt`. Odpowiedź uzasadnij.

Specyfikacja
Wejście: n– liczba naturalna; a – tablica liczb naturalnych

Wyjście: .....

.....

.....

c) [2] Podaj zależność rekurencyjną, która określa złożoność czasową funkcji `pr` w notacji asymptotycznej (przyjmując, że n = j – i +1).

Zależność rekurencyjna
T(1) = 1

T(n) = ... dla n>1, n parzyste

T(n) = ... dla n>1, n nieparzyste

Podaj uzasadnienia do punktów b) i c) na odwrocie kartki.

Imię i nazwisko: ..... **WdI, Kolokwium nr 2**
Indeks: .....

4. **[Wersja 1: 8 punktów]** Chcemy rozmieścić żetony na kwadratowej planszy rozmiaru n × n w taki sposób, że spełnione są jednocześnie poniższe warunki:
   a. w każdej kolumnie znajduje się dokładnie 1 żeton
   b. w każdym wierszu znajduje się dokładnie 1 żeton

   Za umieszczenie żetonu na polu (i, j) planszy musimy zapłacić a[i][j] złotych.
   Napisz funkcję realizującą poniższą specyfikację.

   Wejście: n, s – dodatnie liczby naturalne
   a[n][n] – tablica kwadratowa dodatnich liczb naturalnych

   Wyjście:
   true (lub 1) – jeśli możliwe jest rozmieszczenie żetonów zgodnie z opisanymi powyżej zasadami na planszy rozmiaru n × n, płacąc za to nie więcej niż s złotych.
   false (lub 0) – jeśli NIE jest możliwe rozmieszczenie żetonów zgodnie z opisanymi powyżej zasadami na planszy rozmiaru n × n, płacąc za to nie więcej niż s złotych.

   [Wersja 2: 4 punkty] Rozwiąż powyższe zadanie przy prostszej zasadzie stawiania żetonów: w każdej kolumnie znajduje się dokładnie 1 żeton.

   Postaraj się, aby Twoje rozwiązanie (niezależnie od wybranej wersji) miało jak najmniejszy czas działania.

   Przykłady:
   Dla n=4, s=40 oraz a przedstawionego poniżej:

i, j	**0**	**1**	**2**	**3**
0	10	11	11	11
1	11	10	11	11
2	11	11	10	11
3	11	11	11	10

   funkcja powinna zwrócić true w obu wariantach zadania, ponieważ możemy ustawić żetony na pozycjach (0,0), (1,1), (2,2) i (3,3) płacąc 40 złotych. Natomiast dla n=4, s=40 oraz a przedstawionego poniżej:

i, j	**0**	**1**	**2**	**3**
0	10	10	10	11
1	11	11	11	10
2	11	11	11	11
3	11	11	11	11

   funkcja powinna zwrócić false w wariancie 1 (w wierszach 2 i 3 musimy wybrać cenę 11). Natomiast w wariancie 2 funkcja zwróci wartość true (w każdej kolumnie możemy wybrać pole za które zapłacimy 10 złotych).