SPRAWDZIAN II

Imię i nazwisko:

Nr indeksu:

Nr grupy:

Uwaga! Sprawdzian jest testem wielokrotnego wyboru, gdzie wszystkie możliwe kombinacje odpowiedzi są dopuszczalne (tj. zarówno wszystkie odpowiedzi poprawne, część odpowiedzi poprawna jak i brak odpowiedzi poprawnych). Poprawne odpowiedzi należy zaznaczyć, z lewej strony kartki, symbolem "+". Natomiast symbol "-" jak i brak symbolu przy odpowiedzi oznacza odpowiedź niepoprawną. Pytanie jest uznane za poprawnie rozwiązane (tj. +1pkt) wtedy i tylko wtedy gdy wszystkie jego odpowiedzi zaznaczone są poprawnie. Życzymy powodzenia ...

1. Załóżmy, że ciąg wejściowy dla algorytmu MergeSort składa się z $n = 2^k$ elementów, gdzie $k \in \mathbb{N}^+$, wtedy:
   (a) [+] $T(n) = \Theta(n \lg n)$,
   (b) [–] $S(n) = \Omega(n)$,
   (c) [+] drzewo podziału ciągu wejściowego dla rozważanego algorytmu jest wysokości $k \pm 1$.

2. Co jest warunkiem końcowym poniższego algorytmu rekurencyjnego, gdzie $n \in \mathbb{N}$?

```
int Cos(int n) {
   if (n==0) return 0;
   return Cos(n-1)+n;
}
```

   (a) [–] $Cos(n) = n^2$.
   (b) [+] $Cos(n) = 0 + 1 + 2 + \ldots + (n-1) + n$.
   (c) [–] $n \cdot Cos(n) = O(n^2)$.

3. Rozważmy algorytm rekurencyjny z zadania 2, wtedy:
   (a) [+] złożoność czasowa algorytmu jest co najwyżej rzędu n^2,
   (b) [+] złożoność czasowa algorytmu jest co najmniej rzędu $\sqrt{n}$,
   (c) [–] złożoność pamięciowa algorytmu jest $O(1)$.

4. Co jest warunkiem końcowym poniższego algorytmu rekurencyjnego, jeżeli założymy, że *root* jest dowiązaniem do korzenia pewnego drzewa binarnego?

```
int Cos(TreeNode root) {
   if (root==NULL) return 0;
   else if ((root.left==NULL)&&(root.right==NULL)) return 2;
      else return Cos(root.left)+Cos(root.right);
}
```

   (a) [–] Suma wierzchołków drzewa binarnego o korzeniu *root*.
   (b) [+] Podwojona liczba wierzchołków zewnętrznych drzewa binarnego o korzeniu *root*.
   (c) [–] Wysokość drzewa binarnego o korzeniu *root*.

5. Rozważmy algorytm rekurencyjny z zadania 4, wtedy:
   (a) [+] złożoność czasowa algorytmu jest równa z dokładnością do rzędu funkcji liczbie wierzchołków drzewa wejściowego,
   (b) [–] złożoność czasowa algorytmu jest $O(h)$, gdzie h jest wysokością drzewa wejściowego,
   (c) [–] złożoność pamięciowa algorytmu jest $O(\lg n)$, gdzie n jest liczbą wierzchołków drzewa wejściowego.

6. Dla algorytmu SelectionSort i danych wejściowych rozmiaru n prawdą jest, że:
   (a) [+] $W(n) = \Omega(n \lg n)$, jeżeli miarą jest liczba porównań elementów,
   (b) [+] $W(n) = O(n \lg n)$, jeżeli miarą jest liczba przestawień elementów,
   (c) [+] $S(n) = O(n \lg n)$.

7. Rozważmy ciąg wejściowy dla algorytmu InsertionSort postaci

   $$6, 5, 4, 3, 2, 1,$$

   wtedy:
   (a) [+] porządek elementów po zakończeniu wstawiania elementu 4 jest następujący: $4, 5, 6, 3, 2, 1$,
   (b) [–] porządek elementów po zakończeniu wstawiania elementu 3 jest następujący: $4, 5, 6, 3, 2, 1$,
   (c) [–] w tym przypadku algorytm wykona 17 porównań.

8. Dla algorytmu QuickSort sortowania n-elementowego ciągu wejściowego zapisanego w implementacji rekurencyjnej prawdą jest, że:
   (a) [+] $A(n) = O(n \lg n)$,
   (b) [–] $W(n) = A(n)$,
   (c) [–] $S(n) = O(1)$.

9. Rozważmy algorytm RadixSort, który zastosowano do posortowania n ciągów binarnych długości k, wtedy:
   (a) [–] jeżeli $n = \Theta\left(\sqrt{k}\right)$, to $T(n, k) = O(n)$,
   (b) [+] jeżeli $k = \Theta(\sqrt{n})$, to $T(n, k) = O(n\sqrt{n})$,
   (c) [–] jeżeli $n = \Omega\left(\sqrt{k}\right)$, to $S(n, k) = O(n)$.

10. Rozważmy algorytm CountingSort dla danych wejściowych

    $$1, 1, 4, 3, 2, 1, 1, 0, 2, 1,$$

    wtedy postać tablicy pomocniczej (tablica TMP) po:
    (a) [–] drugiej części algorytmu (zliczanie) jest następująca: $1, 5, 2, 1, 2$,
    (b) [+] trzeciej części algorytmu (sumowanie) jest następująca: $1, 6, 8, 9, 10$,
    (c) [+] czwartej części algorytmu (wypisanie) jest następująca: $0, 1, 6, 8, 9$.

11. Które z poniższych zdań jest zawsze prawdziwe w dziedzinie stosów:
    (a) [–] $empty(s) \Rightarrow \neg empty(pop(push(s, e)))$,
    (b) [–] $\neg empty(s) \Rightarrow top(s) \neq top(push(pop(s), e))$,
    (c) [+] $\neg empty(s) \Rightarrow top(pop(push(s, top(s)))) = top(s)$?

12. Które z poniższych zdań jest zawsze prawdziwe w dziedzinie kolejek:
    (a) [–] $\neg empty\,(q) \Rightarrow first\,(q) \neq first\,(in\,(out\,(q)\,,e))$,
    (b) [+] $out\,(in\,(in\,(q,e)\,,e)) = in\,(out\,(in\,(q,e))\,,e)$,
    (c) [+] $empty\,(q) \Rightarrow empty\,(in\,(q,e)) = false$?
13. Załóżmy, że stos s zawiera n elementów oraz, że wykonujemy jedynie operacje stosowe, wtedy odczytanie elementu x znajdującego się:
    (a) [–] na wysokości $\Theta\left(\sqrt{n}\right)$ względem „dołu stosu”, wymaga wcześniejszego wykonania $\Theta\left(\sqrt{n}\right)$ operacji *pop* na stosie s,
    (b) [+] na wysokości $\Theta\left(\sqrt{n}\right)$ względem „góry stosu”, wymaga wcześniejszego wykonania $\Theta\left(\sqrt{n}\right)$ operacji *pop* na stosie s,
    (c) [–] na wysokości $\Theta\,(n)$ względem „góry stosu”, wymaga wcześniejszego wykonania $O\,(1)$ operacji *pop* na stosie s.
14. Strukturę danych $\langle E \cup L, add, supr, inf, cut\rangle$, gdzie:
    - *add* – dołączyć element na początek struktury, $add : L \times E \rightarrow L$,
    - *supr* – usunąć pierwszy element ze struktury, $supr : L \rightarrow L$,
    - *inf* – podać najmniejszy element ze struktury, $inf : L \rightarrow E$,
    - *cut* – usunąć element najmniejszy i wszystkie elementy dołączone po nim do struktury, $cut : L \rightarrow L$,

    można zaimplementować przy stałej złożoności wykonania operacji, jeżeli:
    (a) [–] użyjemy dwóch tablic statycznych reprezentujących odpowiednio zbiór E oraz L,
    (b) [–] użyjemy dwóch list jednokierunkowych „działających w trybie kolejek”, kolejno kolejki elementów oraz kolejki elementów najmniejszych,
    (c) [+/–] użyjemy dwóch list jednokierunkowych „działających w trybie stosów”, kolejno stosu elementów oraz stosu elementów najmniejszych.
15. Rozważmy algorytm obliczania wartości n-operatorowego wyrażenia arytmetycznego przy użyciu dwóch stosów (stos operatorów oraz stos argumentów), wtedy:
    (a) [–] złożoność czasowa algorytmu jest rzędu $\sqrt{n}$,
    (b) [–] złożoność pamięciowa algorytmu jest rzędu n,
    (c) [+] złożoność pamięciowa algorytmu jest zależna od sposobu nawiasowania wyrażenia i w przypadku optymistycznym jest stała.
16. Goździkowa twierdzi, że Etopiryna:
    (a) jest do nabycia w najbliższej aptece bez kolejki,
    (b) działa w miejscu,
    (c) jest stabilna w przypadku oczekiwanym.