ASD, egzamin – 27 września, 2017

1. (15p) Określ porządek asymptotycznego wzrostu $g \prec g$ (lub $f \approx g$) dla następujących funkcji (dla $n \geqslant 1$; c, d stałe takie, że $4 < c < 5$ i $10 < d < 20$): $cn^d + dn^c$, $2^{d \log_2(\log_2(n))}$, $n(\log_2(n))^c$, $n^2 \log_2(n) + n(\log_2(n))^2$.

2. (15p) Oszacuj z dokładnością do O(.) czas działania algorytmu (wzgl. rozmiaru tablicy n = koniec - poczatek + 1).

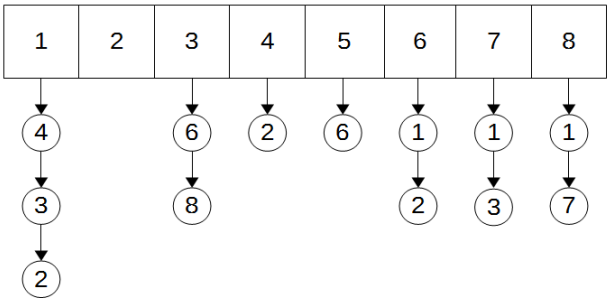
```
Blublu(int *t, int poczatek, int koniec){
  int rozmiar=koniec - poczatek;
  int skok=⌊rozmiar/2⌋;
  if(skok <= 3) return;
  BluBlu(t,poczatek+1, poczatek+skok+1);
  for(i:=0 to i=skok-1)
    print(t[poczatek + i]+t[poczatek+skok+i]);
  if(t[0]>0)
    BluBlu(t,poczatek+2, poczatek+skok+2);
  else
    BluBlu(t,poczatek+3,poczatek+skok+3);
  BluBlu(t,poczatek, poczatek+skok);
}
```

3. (15p) Przesortuj tablicę $t[1 \ldots 7] = \{1, 8, 9, 4, 5, 7, 5\}$ sortowaniem przez kopcowanie. Przyjmij, że kopiec tworzony jest w pętli
`for(i=⌊N/2⌋;i>=1;i-) downheap(t,i,N);`
Wypisz wartość tablicy po każdym przebiegu powyższej pętli (3 tablice). Następnie, wypisz wartości tablicy po każdym umieszczeniu jednego elementu na właściwym miejscu i poprawieniu kopca (6 tablic).

4. (15p) Niech i(x) oznacza instrukcję wstawiania elementu do drzewa BST, d(x) – usuwania. Narysuj drzewo BST, po wykonywaniu ciągu instrukcji: i(10), i(8), i(3), i(15), i(20), i(17), i(30), i(16), i(18), i(22) (5pt). Narysuj drzewo BST powstałe po wykonaniu instrukcji d(10) (przyjmij, że wybierasz podczas usuwania najmniejszy element z prawego podrzewa) (5pt). Rotacjami w lewo i w prawo wypromuj element 16 do korzenia, narysuj kolejne drzewa BST powstałe po przesunięciach (5pt).

5. Narysuj graf skierowany reprezentowany przez poniższą strukturę (5p). Narysuj drzewo wywołań rekurencyjnych funkcji visit_dfs(3) (5pt). Przedstaw w

jakiej kolejności zostaną odwiedzone wierchołki przez visit_dfs(3) (czyli w jakiej kolejności zostaną oznaczone odpowiednie elementy tablicy mark[1..8]) (5p). Przyjmij, że początkowo dla każdego wierzchołka x, mark[x] nie jest równe visited (10p).




```
visit_dfs(x){
   if(mark[x]==visited) return;
   mark[x]=visited;
   Edge *ptr=edges[x];
   while(ptr!=NULL){
      if(mark[ptr->end]!=visited){
         markt[ptr->end]=visited;
         visit_dfs(ptr->end);
      }
      ptr=ptr->next;
   }
}
```

6. (15p) Napisz funkcję *int LiscieModDwa(Node *ptr)*, która zwróci 0 jeśi drzewo ptr ma parzyście wiele liści i 1 w przeciwnym przypadku (czyli funkcja zwraca liczbę liści drzewa modulo 2). Przyjmij, że *Node* to struktura postaci *struct Node{int key; Node *left; Node *right; }*.