STOSTY I KOLEJKI

zad 2

procedure PUSH(S,x,n)

if (S.top<=n)

S.top = S.top + 1

S[S.top] = x

else

write "blad"

return

procedure PUSH(S,x)

if (S.top<=S.length)

S.top = S.top + 1

S[S.top] = x

else

write "blad"

return

zad 3

procedure SKLEJ\_SORT (S1,S2)

S.top=0 // tworzy stos

while not (STACK\_EMPTY(S1))

do PUSH(S,POP(S1))

while not (STACK\_EMPTY(S2))

do PUSH(S,POP(S2))

return

zad 4

procedure ODWROC\_STOS(S)

S1.top=0

S2.top=0

while not (STACK\_EMPTY(S))

do PUSH(S1,POP(S))

while not (STACK\_EMPTY(S1))

do PUSH(S2,POP(S1))

while not (STACK\_EMPTY(S2))

do PUSH(S,POP(S2))

zad 7

procedure MAKE\_QUEUE(Q,S)

Q.head = Q.tail = 1

while not (STACK\_EMPTY(S))

do x = POP(S)

ENQUEUE(Q,x)

Return

zad 5

Q | | | | | | |

Q.head Q.tail

Po ENQUEUE(Q,4)

Q |4| | | | | | |

Po ENQUEUE(Q,1)

Q |4|1| | | | | |

Po ENQUEUE(Q,3)

Q |4|1|3| | | | |

Po DEQUEUE(Q)

Q | |1|3| | | | |

Po DEQUEUE(Q)

Q | | |3| | | | |

Po ENQUEUE(Q,8)

Q | | |3|8| | | |

zad 6

procedure ENQUEUE(Q,x)

if Q.head == Q.tail+1

then return "kolejka jest pelna"

else

Q[Q.tail] = x

if Q.tail == Q.length

then Q.tail = 1

else Q.tail = Q.tail + 1

return

procedure DEQUEUE(Q)

if QUEUE\_EMPTY == true

then return "kolejka jest pusta"

else

x = Q[Q.head]

if Q.head == Q.length

then Q.tail = 1

else Q.head = Q.head + 1

return x

zad 8

procedure ODWROC\_KOLEJKE(Q1)

Q2.head = Q2.tail =1

S.top = 0

while not (QUEUE\_EMPTY(Q1))

do PUSH(S,DEQUEUE(Q1))

while not (STACK\_EMPTY(S))

do ENQUEUE(Q2,POP(S))

return

procedure odw (Q1,Q1.head,tab[1..Q1.tail])

i=0

while not QUEUE\_EMPTY(Q1) do

I=i+1

tab[1]=Q1[Q1.head]

DEQUEUE(Q1)

Q2.head=Q2.tail=1

for j=1 to i do

ENQUEUE(Q2,tab[j])

B zadania domowe

zad 1

function MAX\_STOS(S)

if not STACK\_EMPTY(S)

then max=POP(S)

else return "Stos jest pusty"

while not (STACK\_EMPTY(S))

do pom=POP(S)

if(max<pom)

then max=pom

return max

zad 2

function WSTAW\_DO\_KOLEJKI(S1,S2)

Q.head=Q.tail=1

While not STACK\_EMPTY(S1)

do ENQUEUE(Q,POP(S1))

while not STACK\_EMPTY(S2)

do ENQUEUE(Q,POP(S2))

return Q

LISTY

zad 1

function LENGTH(L)

i = 0

x=L.head

while( x<> NIL) do

x=x.next

i=i+1

return i

zad2

procedure LIST\_INSERT(L,x,k)

i=0

y=L.head

while(y<>NIL) do

if(i==k)

then LIST\_INSERT(L,x)

y=y.next

i=i+1

return

zad3

procedure LIST\_TAIL\_INSERT(L,x)

a=L.head

while(a <> NIL)

do a=a.next

a=x

a.key=x.key

zad 4

procedure ODWORC\_LISTE(L)

preorder klp

inorder lkp

postorder lpk

zad 1

procedure INORDER(T)

p=T.root

if p <> NIL

then INORDER(p.left)

write p.key

INORDER(p.right)

return

procedure POSTORDER(T)

p=T.root

if p <> NIL

then POSTORDER(p.left)

POSTORDER(p.right)

write p.key

return

zad 2

(1,2), (1,3), (2,4), (2,5), (3,6)

1

2 3

4 5 6

preorder : 1, 2, 4, 5, 3, 6

inorder : 4, 2, 5, 1, 6, 3

postorder : 4, 5, 2, 6, 3, 1

zad 3

(1,2), (1,3), (2,4), (2,5), (5,6), (5,7), (7,8),(7,9), (8,10).

1

2 3

4 5

6 7

8 9

10

preorder : 1, 2, 4, 5, 6, 7, 8, 10, 9, 3

inorder : 4, 2, 6, 5, 10, 8, 7, 9, 1, 3,

postorder : 4, 6, 10, 8, 9, 7, 5, 2, 3, 1

zadanie

napisac funkcje zwracajaca sume wszystkich kluczy danego drzewa binarnego

zmienna suma globalna

zamiast wypisania klucza bedzie dodanie klucza do sumy

zadanie 4

A=[4,9,11,5,7,12,32,2,1,10]

4

2 9

1 5 11

7 10 12

32

zadanie 5

function TREE\_MIN(T)

p=T.root

min = p

if p<>NIL

min=TREE\_MIN(p.left)

return min

function TREE\_MAX(T)

p=T.root

max=p

if p<>NIL

max=TREE\_MAX(p.right)

return min

function MIN(T)

x=T.root

while x.left<>NIL

do x=x.left

return x.key

function MAX(T)

x=T.root

while x.right<>NIL

do x=x.right

return x.key

function TREE\_SUM\_KEY(T)

return TREE\_MIN(T) + TREE\_MIN(T)

zad1

25

20 23

14 7 18 19

8 3 2 0

zad 2

2^(h+1) - 1 najwieksza

2^(h) +1

zad 3

A=[27,17,3,16,13,10,1,5,7,12,4,8,9]

1:27

2:17 3:3

4:16 5:13 6:10 7:1

8:5 9:7 10:12 11:4 12:8 13:9

----------------------------------------------------------

1:27

2:17 3:10

4:16 5:13 6:3 7:1

8:5 9:7 10:12 11:4 12:8 13:9

----------------------------------------------------------

1:27

2:17 3:10

4:16 5:13 6:9 7:1

8:5 9:7 10:12 11:4 12:8 13:3

