function union(S1, S2: Set) → Set kamus lokal

i : integer S : Set

algoritma

i traversal [0..s1.length - 1]

S.buffer[i] = S1.buffer[i]

S.length = S1.length

j traversal [0..S2.length - 1]

a <- 0

while S2.buffer[j] != S1.buffer[a]

a += 1

if a = S1.length then

S.buffer[S.length] = S2.buffer[j]

S.length += 1

→ S

function isEmpty(Q: Queue) → boolean kamus lokal

algoritma

→ (Q.idxHead = IDX\_UNDEF) and (Q.idxTail = IDX\_UNDEF)

function isFull(Q: Queue) → boolean kamus lokal

algoritma

→ (Q.idxHead-Q.idxTail = 1) or (Q.idxTail-Q.idxHead = CAPACITY-1)

function length(Q: Queue) → integer kamus lokal

i : integer algoritma

if Q.idxHead > Q.idxTail then

→ CAPACITY - Q.idxHead + Q.idxTail + 1

else

→ Q.idxTail - Q.idxHead + 1

procedure CreateQueue(output Q: Queue) kamus lokal

algoritma

Q.idxHead ← IDX\_UNDEF Q.idxTail ← IDX\_UNDEF

boolean isEmpty(Queue Q){

return (Q.idxHead == IDX\_UNDEF) || (Q.idxTail == IDX\_UNDEF);

}

boolean isFull(Queue Q){

return (Q.idxHead - Q.idxTail == 1) || (Q.idxTail-Q.idxHead==CAPACITY-1);

}

int length (Queue Q) {

if (Q.idxHead > Q.idxTail) {

return (CAPACITY - Q.idxHead + Q.idxTail + 1);

} else {

return (Q.idxTail - Q.idxHead + 1);

}

}

void CreateQueue (Queue\* Q) { (\*Q).idxHead = IDX\_UNDEF; (\*Q).idxTail = IDX\_UNDEF;

}

void enqueue (Queue\* Q, ElType val) {

if !(isFull(\*Q)) {

if (isEmpty(\*Q)){ (\*Q).idxHead = 0;

(\*Q).idxTail = 0; (\*Q).buffer[0] = val;

} else {

if((\*Q).idxTail == CAPACITY-1) { (\*Q).buffer[0] = val; (\*Q).IdxTail = 0;

} else {

(\*Q).IdxTail += 1;

(\*Q).buffer[(\*Q).idxTail] = val;

}

}

}

}

void dequeue (Queue\* Q, ElType\* val) {

\*val = (\*Q).buffer[(\*Q).IdxHead]

if (length(\*Q) == 1) {

CreateQueue(\*Q)

} else if (length(\*Q) == 2) { (\*Q).idxHead = (\*Q).idxTail;

} else {

(\*Q).idxHead = ((\*Q).idxHead+1) % (CAPACITY)

}

}

program MesinBiner kamus

use MESINKAR

CBin : integer procedure STARTBINER

kamus lokal algoritma

START

EndBiner : Boolean

function pangkat(x:integer, y:integer) → integer kamus lokal

i : integer Algoritma

if y = 1 then

→ x

i traversal [1..y] x ← x \* x

→ x procedure ADVBINER

kamus lokal

length, temp, i, try: integer algoritma

length ← 5 {dimisalkan dahulu} try ← 4

while CC!= ‘ ‘ do

if length = 0 then length = 5

temp ← temp \* 32 try ← try + 4

if CC=’1’ then

temp ← pangkat(2, length-1) + temp

algoritma

ADV

length ← length - 1 if length != 0

temp ← temp / pangkat(2, length - 1 + try) Cbin ← temp