***array***

*#include<stdio.h>*

*#include "array.h"*

*/\* \*\*\*\*\*\*\*\*\*\* KONSTRUKTOR \*\*\*\*\*\*\*\*\*\* \*/*

*/\* Konstruktor : create tabel kosong \*/*

*void MakeEmpty (TabInt \*T){*

*(\*T).Neff = 0;*

*};*

*/\* I.S. sembarang \*/*

*/\* F.S. Terbentuk tabel T kosong dengan kapasitas IdxMax-IdxMin+1 \*/*

*/\* \*\*\*\*\*\*\*\*\*\* SELEKTOR \*\*\*\*\*\*\*\*\*\* \*/*

*/\* \*\*\* Banyaknya elemen \*\*\* \*/*

*int NbElmt (TabInt T){*

*return T.Neff;*

*};*

*/\* Mengirimkan banyaknya elemen efektif tabel \*/*

*/\* Mengirimkan nol jika tabel kosong \*/*

*/\* \*\*\* Daya tampung container \*\*\* \*/*

*int MaxNbEl (TabInt T){*

*return IdxMax;*

*};*

*/\* Mengirimkan maksimum elemen yang dapat ditampung oleh tabel \*/*

*/\* \*\*\* Selektor INDEKS \*\*\* \*/*

*IdxType GetFirstIdx (TabInt T){*

*return IdxMin;*

*};*

*/\* Prekondisi : Tabel T tidak kosong \*/*

*/\* Mengirimkan indeks elemen pertama \*/*

*IdxType GetLastIdx (TabInt T){*

*return NbElmt(T);*

*};*

*/\* Prekondisi : Tabel T tidak kosong \*/*

*/\* Mengirimkan indeks elemen terakhir \*/*

*/\* \*\*\* Menghasilkan sebuah elemen \*\*\* \*/*

*ElType GetElmt (TabInt T, IdxType i){*

*return T.TI[i];*

*};*

*/\* Prekondisi : Tabel tidak kosong, i antara FirstIdx(T)..LastIdx(T) \*/*

*/\* Mengirimkan elemen tabel yang ke-i \*/*

*/\* \*\*\* Selektor SET : Mengubah nilai TABEL dan elemen tabel \*\*\* \*/*

*/\* Untuk type private/limited private pada bahasa tertentu \*/*

*void SetTab (TabInt Tin, TabInt \*Tout){*

*int i ;*

*for(i = IdxMin;i<=IdxMax;i++){*

*(\*Tout).TI[i] = Tin.TI[i];*

*}*

*(\*Tout).Neff = Tin.Neff;*

*};*

*/\* I.S. Tin terdefinisi, sembarang \*/*

*/\* F.S. Tout berisi salinan Tin \*/*

*/\* Assignment THsl -> Tin \*/*

*void SetEl (TabInt \*T, IdxType i, ElType v){*

*(\*T).TI[i] = v;*

*if (i == GetLastIdx(\*T) + 1)*

*{*

*(\*T).Neff ++;*

*}*

*};*

*/\* I.S. T terdefinisi, sembarang \*/*

*/\* F.S. Elemen T yang ke-i bernilai v \*/*

*/\* Mengeset nilai elemen tabel yang ke-i sehingga bernilai v \*/*

*void SetNeff (TabInt \*T, IdxType N){*

*(\*T).Neff = N;*

*};*

*/\* I.S. T terdefinisi, sembarang \*/*

*/\* F.S. Nilai indeks efektif T bernilai N \*/*

*/\* Mengeset nilai indeks elemen efektif sehingga bernilai N \*/*

*/\* \*\*\*\*\*\*\*\*\*\* Test Indeks yang valid \*\*\*\*\*\*\*\*\*\* \*/*

*boolean IsIdxValid (TabInt T, IdxType i){*

*return ((i >= IdxMin) && (i <= IdxMax));*

*};*

*/\* Prekondisi : i sembarang \*/*

*/\* Mengirimkan true jika i adalah indeks yang valid utk ukuran tabel \*/*

*/\* yaitu antara indeks yang terdefinisi utk container\*/*

*boolean IsIdxEff (TabInt T, IdxType i){*

*return ((i >= GetFirstIdx(T)) && (i <= GetLastIdx(T)));*

*};*

*/\* Prekondisi : i sembarang\*/*

*/\* Mengirimkan true jika i adalah indeks yang terdefinisi utk tabel \*/*

*/\* yaitu antara FirstIdx(T)..LastIdx(T) \*/*

*/\* \*\*\*\*\*\*\*\*\*\* TEST KOSONG/PENUH \*\*\*\*\*\*\*\*\*\* \*/*

*/\* \*\*\* Test tabel kosong \*\*\* \*/*

*boolean IsEmpty (TabInt T){*

*return (NbElmt(T) == 0);*

*};*

*/\* Mengirimkan true jika tabel T kosong, mengirimkan false jika tidak \*/*

*/\* \*\*\* Test tabel penuh \*\*\* \*/*

*boolean IsFull (TabInt T){*

*return (NbElmt(T) == MaxNbEl(T));*

*};*

*/\* Mengirimkan true jika tabel T penuh, mengirimkan false jika tidak \*/*

*/\* \*\*\*\*\*\*\*\*\*\* BACA dan TULIS dengan INPUT/OUTPUT device \*\*\*\*\*\*\*\*\*\* \*/*

*void TulisIsi (TabInt T){*

*int i;*

*if(IsEmpty(T)){*

*printf("Tabel kosong\n");*

*}*

*else{*

*for(i=GetFirstIdx(T);i<=GetLastIdx(T);i++){*

*printf("%d:%d\n",i,T.TI[i]);*

*}*

*}*

*};*

*/\* Proses : Menuliskan isi tabel dengan traversal \*/*

*/\* I.S. T boleh kosong \*/*

*/\* F.S. Jika T tidak kosong : indeks dan elemen tabel ditulis berderet ke bawah \*/*

*/\* Jika isi tabel [1,2,3] maka akan diprint*

*0:1*

*1:2*

*2:3*

*\*/*

*/\* Jika T kosong : Hanya menulis "Tabel kosong" \*/*

*/\* \*\*\*\*\*\*\*\*\*\* OPERATOR ARITMATIKA \*\*\*\*\*\*\*\*\*\* \*/*

*/\* \*\*\* Aritmatika tabel : Penjumlahan, pengurangan, perkalian, ... \*\*\* \*/*

*TabInt PlusTab (TabInt T1, TabInt T2){*

*int i;*

*TabInt THsl;*

*MakeEmpty(&THsl);*

*THsl.Neff = T1.Neff;*

*for(i=GetFirstIdx(T1);i<=GetLastIdx(T1);i++){*

*SetEl(&THsl, i, GetElmt(T1, i) + GetElmt(T2, i));*

*}*

*return(THsl);*

*};*

*/\* Prekondisi : T1 dan T2 berukuran sama dan tidak kosong \*/*

*/\* Mengirimkan T1 + T2 \*/*

*TabInt MinusTab (TabInt T1, TabInt T2){*

*int i;*

*TabInt THsl;*

*MakeEmpty(&THsl);*

*THsl.Neff = T1.Neff;*

*for(i=GetFirstIdx(T1);i<=GetLastIdx(T1);i++){*

*SetEl(&THsl, i, GetElmt(T1, i) - GetElmt(T2, i));*

*}*

*return(THsl);*

*};*

*/\* Prekondisi : T1 dan T2 berukuran sama dan tidak kosong \*/*

*/\* Mengirimkan T1 - T2 \*/*

*/\* \*\*\*\*\*\*\*\*\*\* NILAI EKSTREM \*\*\*\*\*\*\*\*\*\* \*/*

*ElType ValMax (TabInt T){*

*int i;*

*ElType max;*

*max = T.TI[GetFirstIdx(T)];*

*for (i = (GetFirstIdx(T)+1); i <= (GetLastIdx(T)); i++)*

*{*

*if (max < T.TI[i])*

*{*

*max = T.TI[i];*

*}*

*}*

*return(max);*

*};*

*/\* Prekondisi : Tabel T tidak kosong \*/*

*/\* Mengirimkan nilai maksimum tabel \*/*

*ElType ValMin (TabInt T){*

*int i;*

*ElType min;*

*min = T.TI[GetFirstIdx(T)];*

*for (i = (GetFirstIdx(T)+1); i <= (GetLastIdx(T)); i++)*

*{*

*if (min > T.TI[i])*

*{*

*min = T.TI[i];*

*}*

*}*

*return(min);*

*};*

*/\* Prekondisi : Tabel T tidak kosong \*/*

*/\* Mengirimkan nilai minimum tabel \*/*

*/\* \*\*\* Mengirimkan indeks elemen bernilai ekstrem \*\*\* \*/*

*IdxType IdxMaxTab (TabInt T){*

*int i;*

*for (i = (GetFirstIdx(T)+1); i <= (GetLastIdx(T)); i++){*

*if (T.TI[i] == ValMax(T)){*

*return(i);*

*}*

*}*

*};*

*/\* Prekondisi : Tabel T tidak kosong \*/*

*/\* Mengirimkan indeks i dengan elemen ke-i adalah nilai maksimum pada tabel \*/*

*IdxType IdxMinTab (TabInt T){*

*int i;*

*for (i = (GetFirstIdx(T)+1); i <= (GetLastIdx(T)); i++){*

*if (T.TI[i] == ValMin(T)){*

*return(i);*

*}*

*}*

*};*

*/\* Prekondisi : Tabel tidak kosong \*/*

*/\* Mengirimkan indeks i \*/*

*/\* dengan elemen ke-i nilai minimum pada tabel \*/*

**Arraydin**

*#include* <stdio.h>

*#include* "arraydin.h"

*#include* <stdlib.h>

*/\*\**

*\* Konstruktor*

*\* I.S. sembarang*

*\* F.S. Terbentuk ArrayDin kosong dengan ukuran InitialSize*

*\*/*

ArrayDin MakeArrayDin(){

*// ArrayDin arr;*

*// arr.A = (ElType \*) malloc (InitialSize \* sizeof ElType);*

    ArrayDin arr;

    arr.A = (ElType \*) malloc ((InitialSize) \* sizeof (ElType));

    arr.Capacity = InitialSize;

    arr.Neff = 0;

*return* arr;

}

*/\*\**

*\* Destruktor*

*\* I.S. ArrayDin terdefinisi*

*\* F.S. array->A terdealokasi*

*\*/*

void DeallocateArrayDin(ArrayDin \*array){

    free((\*array).A);

}

*/\*\**

*\* Fungsi untuk mengetahui apakah suatu array kosong.*

*\* Prekondisi: array terdefinisi*

*\*/*

boolean IsEmpty(ArrayDin array){

*return* array.Neff == 0;

}

*/\*\**

*\* Fungsi untuk mendapatkan banyaknya elemen efektif array, 0 jika tabel kosong.*

*\* Prekondisi: array terdefinisi*

*\*/*

int Length(ArrayDin array){

*return* array.Neff;

}

*/\*\**

*\* Mengembalikan elemen array L yang ke-I (indeks lojik).*

*\* Prekondisi: array tidak kosong, i di antara 0..Length(array).*

*\*/*

ElType Get(ArrayDin array, IdxType i){

*return* array.A[i];

}

*/\*\**

*\* Fungsi untuk mendapatkan kapasitas yang tersedia.*

*\* Prekondisi: array terdefinisi*

*\*/*

int GetCapacity(ArrayDin array){

*return* array.Capacity;

}

*/\*\**

*\* Fungsi untuk menambahkan elemen baru di index ke-i*

*\* Prekondisi: array terdefinisi, i di antara 0..Length(array).*

*\*/*

void InsertAt(ArrayDin \*array, ElType el, IdxType i){

    int j;

*if* ((\*array).Neff < (\*array).Capacity)

    {

*for* (j = (\*array).Neff; j >= i + 1; j--)

        {

            (\*array).A[j] = (\*array).A[j - 1];

        }

        (\*array).A[i] = el;

        (\*array).Neff += 1;

    }

*else*

    {

        ElType \*temparray = (ElType \*)malloc((\*array).Capacity\* sizeof(ElType));

*for* (j = 0; j < (\*array).Capacity; j++)

        {

            temparray[j] = (\*array).A[j];

        }

        free((\*array).A);

        (\*array).A = (ElType \*)malloc(((\*array).Capacity \* 2 ) \*sizeof(ElType));

*for* (j = 0; j < i; j++)

        {

            (\*array).A[j] = temparray[j];

        }

        (\*array).A[i] = el;

*for* (j = i; j < (\*array).Capacity; j++)

        {

            (\*array).A[j + 1] = temparray[j];

        }

        free(temparray);

        (\*array).Capacity \*= 2;

        (\*array).Neff += 1;

    }

}

*/\*\**

*\* Fungsi untuk menambahkan elemen baru di akhir array.*

*\* Prekondisi: array terdefinisi*

*\*/*

void InsertLast(ArrayDin \*array, ElType el){

    InsertAt((array),el,(\*array).Neff);

}

*/\*\**

*\* Fungsi untuk menambahkan elemen baru di awal array.*

*\* Prekondisi: array terdefinisi*

*\*/*

void InsertFirst(ArrayDin \*array, ElType el){

   InsertAt((array),el,0);

}

*/\*\**

*\* Fungsi untuk menghapus elemen di index ke-i ArrayDin*

*\* Prekondisi: array terdefinisi, i di antara 0..Length(array).*

*\*/*

void DeleteAt(ArrayDin \*array, IdxType i){

    int j;

    ElType e = Get(\*array, i);

*if* ((\*array).Neff > ((\*array).Capacity / 4))

    {

        (\*array).Neff -= 1;

*for* (j = i; j < (\*array).Neff; j++)

        {

            (\*array).A[j] = (\*array).A[j + 1];

        }

    }

*else*

    {

        int Bcapacity = InitialSize;

*if* (Bcapacity < ((\*array).Capacity / 2)) Bcapacity = (\*array).Capacity / 2;

        ElType \*temparray = (ElType \*)malloc(Bcapacity \* sizeof(ElType));

*for* (j = 0; j < i; j++)

        {

            temparray[j] = (\*array).A[j];

        }

        (\*array).Neff -= 1;

*for* (j = i; j < (\*array).Neff; j++)

        {

            temparray[j] = (\*array).A[j + 1];

        }

        free((\*array).A);

        (\*array).A = (ElType \*)malloc(Bcapacity \* sizeof(ElType));

*for* (j = 0; j < (\*array).Neff; j++)

        {

            (\*array).A[j] = temparray[j];

        }

        (\*array).Capacity = Bcapacity;

        free(temparray);

    }

}

*/\*\**

*\* Fungsi untuk menghapus elemen terakhir ArrayDin*

*\* Prekondisi: array tidak kosong*

*\*/*

void DeleteLast(ArrayDin \*array){

    DeleteAt(array,(\*array).Neff-1);

}

*/\*\**

*\* Fungsi untuk menghapus elemen pertama ArrayDin*

*\* Prekondisi: array tidak kosong*

*\*/*

void DeleteFirst(ArrayDin \*array){

    DeleteAt(array,0);

}

*/\*\**

*\* Fungsi untuk melakukan print suatu ArrayDin.*

*\* Print dilakukan dengan format: [elemen-1, elemen-2, ..., elemen-n]*

*\* dan diakhiri newline.*

*\* Prekondisi: array terdefinisi*

*\*/*

void PrintArrayDin(ArrayDin array){

    int j;

    printf("[");

*for* (j=0;j<(array).Neff;j++){

        printf("%d",(array).A[j]);

*if* (j!= (array).Neff-1){

            printf(", ");

        }

    }

    printf("]\n");

}

*/\*\**

*\* Fungsi untuk melakukan reverse suatu ArrayDin.*

*\* Prekondisi: array terdefinisi*

*\*/*

void ReverseArrayDin(ArrayDin \*array){

    int j;

    ElType \*temp = (ElType\*) malloc ((\*array).Capacity \* sizeof(ElType));

*for* (j = 0; j < (\*array).Neff; j++) {

        temp[j] = (\*array).A[(\*array).Neff - (1+j)];

    }

*for* (j = 0; j < (\*array).Neff; j++) {

        (\*array).A[j] = temp[j];

    }

    free(temp);

}

*/\*\**

*\* Fungsi untuk melakukan copy suatu ArrayDin.*

*\* Prekondisi: array terdefinisi*

*\*/*

ArrayDin CopyArrayDin(ArrayDin array){

    ArrayDin coparr;

    IdxType j;

    coparr.A = (ElType\*) malloc (array.Capacity \* sizeof(ElType));

    coparr.Capacity = array.Capacity; coparr.Neff = array.Neff;

*for* (j = 0; j < coparr.Neff; j++) {

        coparr.A[j] = array.A[j];

    }

*return* coparr;

}

*/\*\**

*\* Fungsi untuk melakukan search suatu ArrayDin.*

*\* Index pertama yang ditemukan akan dikembalikan.*

*\* Jika tidak ditemukan, akan mengembalikan -1.*

*\* Prekondisi: array terdefinisi*

*\*/*

IdxType SearchArrayDin(ArrayDin array, ElType el){

    int j =0;

*while* (((array).A[j] != el) && (j <= (array).Neff)) {

        j++;

    }

*if* (j > (array).Neff) {

        j = -1;

    }

*return* j;

}

**ARRAY MHS**

*#include<stdio.h>*

*#include "arrayMhs.h"*

*#include <stdlib.h>*

*/\* \*\*\*\*\*\*\*\*\*\* KONSTRUKTOR ARRAY \*\*\*\*\*\*\*\*\*\* \*/*

*/\* Konstruktor : create tabel kosong \*/*

*void MakeEmpty (TabMhs \*T){*

*(\*T).Neff = 0;*

*}*

*int NbElmt (TabMhs T){*

*return T.Neff;*

*}*

*/\* Mengirimkan banyaknya elemen efektif tabel \*/*

*/\* Mengirimkan nol jika tabel kosong \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// return T.Neff;*

*/\* \*\*\* Daya tampung container \*\*\* \*/*

*int MaxNbEl (TabMhs T){*

*return IdxMax - IdxMin + 1;*

*}*

*/\* Mengirimkan maksimum elemen yang dapat ditampung oleh tabel \*/*

*/\* \*\*\* Selektor INDEKS \*\*\* \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// return IdxMax - IdxMin + 1;*

*IdxType GetFirstIdx (TabMhs T){*

*return IdxMin;*

*}*

*/\* Prekondisi : Tabel T tidak kosong \*/*

*/\* Mengirimkan indeks elemen pertama \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// return IdxMin;*

*IdxType GetLastIdx (TabMhs T){*

*return T.Neff;*

*}*

*/\* Prekondisi : Tabel T tidak kosong \*/*

*/\* Mengirimkan indeks elemen terakhir \*/*

*/\* \*\*\* Menghasilkan sebuah elemen \*\*\* \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// return T.Neff;*

*ElType GetElmt (TabMhs T, IdxType i){*

*return T.TI[i];*

*}*

*/\* Prekondisi : Tabel tidak kosong, i antara FirstIdx(T)..LastIdx(T) \*/*

*/\* Mengirimkan elemen tabel yang ke-i \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// return T.TI[i];*

*/\* \*\*\* Selektor SET : Mengubah nilai TABEL dan elemen tabel \*\*\* \*/*

*/\* Untuk type private/limited private pada bahasa tertentu \*/*

*void SetTab (TabMhs Tin, TabMhs \*Tout){*

*(\*Tout).Neff = Tin.Neff;*

*for(int i = IdxMin; i <= Tin.Neff; i++){*

*(\*Tout).TI[i] = Tin.TI[i];*

*}*

*}*

*/\* I.S. Tin terdefinisi, sembarang \*/*

*/\* F.S. Tout berisi salinan Tin \*/*

*/\* Assignment THsl -> Tin \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// (\*Tout).Neff = Tin.Neff;*

*// for(int i = IdxMin; i <= Tin.Neff; i++){*

*//     (\*Tout).TI[i] = Tin.TI[i];*

*// }*

*void SetEl (TabMhs \*T, IdxType i, ElType v){*

*(\*T).TI[i] = v;*

*if ((\*T).Neff < i){*

*(\*T).Neff = i;*

*}*

*}*

*/\* I.S. T terdefinisi, sembarang \*/*

*/\* F.S. Elemen T yang ke-i bernilai v \*/*

*/\* Mengeset nilai elemen tabel yang ke-i sehingga bernilai v \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// (\*T).TI[i] = v;*

*// if ((\*T).Neff < i){*

*//     (\*T).Neff = i;*

*// }*

*void SetNeff (TabMhs \*T, IdxType N){*

*(\*T).Neff = N;*

*}*

*/\* I.S. T terdefinisi, sembarang \*/*

*/\* F.S. Nilai indeks efektif T bernilai N \*/*

*/\* Mengeset nilai indeks elemen efektif sehingga bernilai N \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// (\*T).Neff = N;*

*/\* \*\*\*\*\*\*\*\*\*\* Test Indeks yang valid \*\*\*\*\*\*\*\*\*\* \*/*

*boolean IsIdxValid (TabMhs T, IdxType i){*

*if ((i >= IdxMin) && (i <= IdxMax)){*

*return true;*

*}*

*else {*

*return false;*

*}*

*}*

*/\* Prekondisi : i sembarang \*/*

*/\* Mengirimkan true jika i adalah indeks yang valid utk ukuran tabel \*/*

*/\* yaitu antara indeks yang terdefinisi utk container\*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// if ((i >= IdxMin) && (i <= IdxMax)){*

*//     return true;*

*// } else {*

*//     return false;*

*// }*

*boolean IsIdxEff (TabMhs T, IdxType i){*

*if ((i >= IdxMin) && (i <= T.Neff)){*

*return true;*

*} else {*

*return false;*

*}*

*}*

*/\* Prekondisi : i sembarang\*/*

*/\* Mengirimkan true jika i adalah indeks yang terdefinisi utk tabel \*/*

*/\* yaitu antara FirstIdx(T)..LastIdx(T) \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// if ((i >= IdxMin) && (i <= T.Neff)){*

*//     return true;*

*// } else {*

*//     return false;*

*// }*

*/\* \*\*\*\*\*\*\*\*\*\* TEST KOSONG/PENUH \*\*\*\*\*\*\*\*\*\* \*/*

*/\* \*\*\* Test tabel kosong \*\*\* \*/*

*boolean IsEmpty (TabMhs T){*

*if (T.Neff > 0){*

*return false;*

*} else {*

*return true;*

*}*

*}*

*/\* Mengirimkan true jika tabel T kosong, mengirimkan false jika tidak \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// if (T.Neff > 0){*

*//     return false;*

*// } else {*

*//     return true;*

*// }*

*/\* \*\*\* Test tabel penuh \*\*\* \*/*

*boolean IsFull (TabMhs T){*

*if (T.Neff > 0){*

*return false;*

*} else {*

*return true;*

*}*

*}*

*/\* Mengirimkan true jika tabel T penuh, mengirimkan false jika tidak \*/*

*/\* Untuk membantu pengerjaan praktikum, silahkan gunakan kode berikut untuk fungsi ini \*/*

*// if (T.Neff == IdxMax){*

*//     return true;*

*// } else {*

*//     return false;*

*// }*

*/\* \*\*\*\*\*\*\*\*\*\* BACA dan TULIS dengan INPUT/OUTPUT device \*\*\*\*\*\*\*\*\*\* \*/*

*void TulisIsi (TabMhs T){*

*if(IsEmpty(T)){*

*printf("Tabel kosong\n");*

*}*

*else{*

*int i;*

*for(i=1;i<=NbElmt(T);i++){*

*printf("%s | %s | %.2f\n",T.TI[i].nama,T.TI[i].nim,T.TI[i].nilai);*

*}*

*}*

*}*

*/\* Proses : Menuliskan isi tabel dengan traversal \*/*

*/\* I.S. T boleh kosong \*/*

*/\* F.S. Jika T tidak kosong : indeks dan elemen tabel ditulis berderet ke bawah \*/*

*/\* Contoh print tabel*

*Test | 12321123 | 3.61*

*Ini Juga Test | 12321124 | 3.21*

*Test Lagi | 12321125 | 3.11*

*\*/*

*/\* Jika T kosong : Hanya menulis "Tabel kosong" \*/*

*/\* \*\*\*\*\*\*\*\*\*\* KONSTRUKTOR MAHASISWA \*\*\*\*\*\*\*\*\*\* \*/*

*ElType MakeMahasiswa (char\* Nama, char\* NIM, float Nilai){*

*ElType M;*

*M.nama = Nama;*

*M.nim = NIM;*

*M.nilai = Nilai;*

*return M;*

*}*

*/\* Membentuk sebuah Mahasiswa dari komponen-komponennya \*/*

*/\* I.S. Nama, NIM, Nilai terdefinisi \*/*

*/\* F.S. Mahasiswa M terbentuk dengan Nama, NIM, Nilai yang sesuai \*/*

*/\* \*\*\*\*\*\*\*\*\*\* OPERATOR STATISTIK MAHASISWA \*\*\*\*\*\*\*\*\*\* \*/*

*float RataRata (TabMhs T){*

*float sum=0;*

*int i;*

*for(i=1;i<=NbElmt(T);i++){*

*sum+=T.TI[i].nilai;*

*}*

*return(sum/NbElmt(T));*

*}*

*/\* I.S. Tabel T tidak kosong \*/*

*/\* F.S. Menghasilkan nilai rata-rata dari elemen tabel \*/*

*float Max (TabMhs T){*

*float max = T.TI[1].nilai;*

*int i;*

*for(i=1;i<=NbElmt(T);i++){*

*if (T.TI[i].nilai > max){*

*max = T.TI[i].nilai;*

*}*

*}*

*return max;*

*}*

*/\* I.S. Tabel T tidak kosong \*/*

*/\* F.S. Menghasilkan nilai maksimum dari elemen tabel \*/*

*float Min (TabMhs T){*

*float min = T.TI[1].nilai;*

*int i;*

*for(i=1;i<=NbElmt(T);i++){*

*if (T.TI[i].nilai < min){*

*min = T.TI[i].nilai;*

*}*

*}*

*return min;*

*}*

*/\* I.S. Tabel T tidak kosong \*/*

*/\* F.S. Menghasilkan nilai minimum dari elemen tabel \*/*

*char \*MaxNama (TabMhs T){*

*int i,count=0;*

*for(i=1;i<=NbElmt(T);i++){*

*if(T.TI[i].nilai == Max(T)){*

*count++;*

*}*

*}*

*if(count == 1){*

*return(T.TI[i].nama);*

*}*

*}*

*/\* I.S. Tabel T tidak kosong \*/*

*/\* F.S. Menghasilkan nama mahasiswa dengan nilai tertinggi \*/*

*/\* Jika ada lebih dari satu mahasiswa dengan nilai tertinggi, maka yang diambil yang nim lebih rendah \*/*

*char \*MinNama (TabMhs T){*

*int i,count=0;*

*for(i=1;i<=NbElmt(T);i++){*

*if(T.TI[i].nilai == Min(T)){*

*count++;*

*}*

*}*

*if(count == 1){*

*return(T.TI[i].nama);*

*}*

*}*

*/\* I.S. Tabel T tidak kosong \*/*

*/\* F.S. Menghasilkan nama mahasiswa dengan nilai terendah \*/*

*/\* Jika ada lebih dari satu mahasiswa dengan nilai terendah, maka yang diambil yang nim lebih rendah \*/*

*int JumlahJurusan (TabMhs T, char\* Jurusan){*

*int i,j,count,check;*

*char nim\_titip[3];*

*count = 0;*

*for (i = (GetFirstIdx(T)); i <= (GetLastIdx(T)); i++)*

*{*

*for (j = 0; j < 3;j++)*

*{*

*nim\_titip[j] = T.TI[i].nim[j];*

*}*

*check = strcmp(nim\_titip, \*Jurusan);*

*if (check == 0)*

*{*

*count++;*

*}*

*}*

*return count;*

*}*

*/\* I.S. Tabel T tidak kosong \*/*

*/\* F.S. Menghasilkan jumlah mahasiswa yang berasal dari jurusan Jurusan, yang berbentuk 3 angka (contoh: 182) \*/*

*/\* Jika tidak ada mahasiswa yang berasal dari jurusan Jurusan, maka menghasilkan 0 \*/*

**GARIS**

*#include* <stdio.h>

*#include* "garis.h"

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/*

*/\* DEFINISI PRIMITIF                                                 \*/*

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/*

*/\* \*\*\* Konstruktor membentuk GARIS \*\*\* \*/*

void MakeGARIS(POINT P1, POINT P2, GARIS \*L){

    PAwal(\*L) = P1;

    PAkhir(\*L) = P2;

}

*/\* I.S. P1 dan P2 terdefinisi \*/*

*/\* F.S. L terdefinisi dengan L.PAw = P1 dan L.PAkh = P2 \*/*

*/\* Membentuk sebuah L dari komponen-komponennya \*/*

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/*

*/\* KELOMPOK INTERAKSI DENGAN I/O DEVICE, BACA/TULIS                  \*/*

*/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/*

void BacaGARIS(GARIS \*L){

    POINT P1,P2;

    BacaPOINT(&P1);

    BacaPOINT(&P2);

*while* (EQ(P1,P2)){

        printf("Garis tidak valid\n");

        BacaPOINT(&P1);

        BacaPOINT(&P2);

    }

    MakeGARIS(P1,P2,L);

}

*/\* I.S. sembarang \*/*

*/\* F.S. L terdefinisi sebagai GARIS yang valid \*/*

*/\* Proses : mengulangi membaca dua buah nilai P1 dan P2 sehingga dapat*

*membentuk GARIS yang valid*

*GARIS disebut valid jika titik awal tidak sama dengan titik akhirnya. \*/*

*/\* Jika pembacaan garis tidak valid, diberikan pesan kesalahan:*

*"Garis tidak valid" \*/*

*/\* Contoh:*

*2 3*

*2 3*

*Garis tidak valid*

*2 3*

*3 3*

*--> terbentuk garis L dengan L.PAw = <2,3> dan L.PAkh = <3,3> \*/*

void TulisGARIS(GARIS L){

    printf("(");

    TulisPOINT(PAwal(L));

    printf(",");

    TulisPOINT(PAkhir(L));

    printf(")");

}

*/\* I.S. L terdefinisi \*/*

*/\* F.S. L tertulis di layar dengan format "((x1,y1),(x2,y2))"*

*tanpa tambahan karakter apa pun di depan, di tengah, atau di belakang,*

*termasuk spasi dan enter \*/*

*/\* Output X dan Y harus dituliskan dalam bilangan riil dengan 2 angka di belakang koma. \*/*

*/\* \*\*\* Kelompok operasi garis \*\*\* \*/*

float PanjangGARIS(GARIS L){

*return* (Panjang(PAwal(L),PAkhir(L)));

}

*/\* Menghitung panjang garis L : berikan rumusnya \*/*

float Gradien(GARIS L){

*return*((Ordinat(PAkhir(L))-Ordinat(PAwal(L)))/(Absis(PAkhir(L))-Absis(PAwal(L))));

}

*/\* Mengembalikan Gradien (m) dari L \*/*

*/\* Gradien garis yang melalui (x1,y1) dan (x2,y2) adalah: (y2-y1)/(x2-x1) \*/*

*/\* Prekondisi : x1 != x2 \*/*

*/\* \*\*\* Kelompok predikat \*\*\* \*/*

boolean IsTegakLurus(GARIS L1, GARIS L2){

*return* (Gradien(L1)\*Gradien(L2) == -1);

}

*/\* Menghasilkan true jika L1 tegak lurus terhadap L2 \*/*

*/\* Dua garis saling tegak lurus jika hasil perkalian kedua gradiennya = -1 \*/*

boolean IsSejajar(GARIS L1, GARIS L2){

*return* (Gradien(L1)==Gradien(L2));

}

*/\* Menghasilkan true jika L "sejajar" terhadap L1 \*/*

*/\* Dua garis saling sejajar jika memiliki gradien yang sama \*/*

**LIST**

*#include <stdio.h>*

*#include "list.h"*

*/\* \*\*\*\*\*\*\*\*\*\* KONSTRUKTOR \*\*\*\*\*\*\*\*\*\* \*/*

*/\* Konstruktor: create list kosong \*/*

*List MakeList(){*

*List L;*

*int i;*

*for(i=0;i<MaxEl;i++){*

*L.A[i] = Mark;*

*}*

*return L;*

*}*

*/\* I.S. sembarang \*/*

*/\* F.S. Terbentuk list L kosong dengan kapasitas MaxEl \*/*

*/\* \*\*\*\*\*\*\*\*\*\* TEST KOSONG/PENUH \*\*\*\*\*\*\*\*\*\* \*/*

*/\* \*\*\* Test list kosong \*\*\* \*/*

*boolean IsEmpty(List L){*

*return(L.A[0] == Mark);*

*}*

*/\* Mengirimkan true jika list L kosong, mengirimkan false jika tidak \*/*

*/\* \*\*\* Menghasilkan sebuah elemen \*\*\* \*/*

*ElType Get(List L, IdxType i){*

*return(L.A[i]);*

*}*

*/\* Prekondisi : list tidak kosong, i antara FirstIdx(T)..LastIdx(T) \*/*

*/\* Mengirimkan elemen list yang ke-i \*/*

*/\* \*\*\* Selektor SET : Mengubah nilai list dan elemen list \*\*\* \*/*

*void Set(List \*L, IdxType i, ElType v){*

*(\*L).A[i] = v;*

*}*

*/\* I.S. T terdefinisi, sembarang \*/*

*/\* F.S. Elemen T yang ke-i bernilai v \*/*

*/\* \*\*\*\*\*\*\*\*\*\* SELEKTOR \*\*\*\*\*\*\*\*\*\* \*/*

*/\* \*\*\* Banyaknya elemen \*\*\* \*/*

*int Length(List L){*

*int count,i;*

*count = 0;*

*for(i = 0;i < MaxEl ; i++){*

*if(L.A[i] != Mark){*

*count ++;*

*}*

*}*

*return count;*

*}*

*/\* Mengirimkan banyaknya elemen efektif list \*/*

*/\* Mengirimkan nol jika list kosong \*/*

*/\* \*\*\* Selektor INDEKS \*\*\* \*/*

*IdxType FirstIdx(List L){*

*return 0;*

*/\* Prekondisi : list L tidak kosong \*/*

*/\* Mengirimkan indeks elemen pertama \*/*

*}*

*IdxType LastIdx(List L){*

*int i;*

*for(i=0;i <  MaxEl;i++){*

*if(L.A[i] == Mark){*

*return(i-1);*

*}*

*}*

*}*

*/\* Prekondisi : list L tidak kosong \*/*

*/\* Mengirimkan indeks elemen terakhir \*/*

*/\* \*\*\*\*\*\*\*\*\*\* Test Indeks yang valid \*\*\*\*\*\*\*\*\*\* \*/*

*boolean IsIdxValid (List L, IdxType i){*

*return((i >= 0) && (i< MaxEl));*

*}*

*/\* Prekondisi : i sembarang \*/*

*/\* Mengirimkan true jika i adalah indeks yang valid utk ukuran list \*/*

*/\* yaitu antara indeks yang terdefinisi untuk container\*/*

*boolean IsIdxEff (List L, IdxType i){*

*return((i >= FirstIdx(L)) && (i <= LastIdx(L)));*

*}*

*/\* Prekondisi : i sembarang\*/*

*/\* Mengirimkan true jika i adalah indeks yang terdefinisi utk list \*/*

*/\* yaitu antara FirstIdx(L)..LastIdx(L) \*/*

*/\* \*\*\*\*\*\*\*\*\*\* Operasi-operasi \*\*\*\*\*\*\*\*\*\* \*/*

*boolean Search(List L, ElType X){*

*int i,cek;*

*cek = 0;*

*for(i= 0 ; i < MaxEl ; i++){*

*if(L.A[i] == X){*

*cek = 1;*

*}*

*}*

*return(cek == 1);*

*}*

*/\* Prekondisi : X sembarang \*/*

*/\* Mengirimkan true jika terdapat elemen X di dalam list \*/*

*/\* yaitu antara FirstIdx(L)..LastIdx(L) \*/*

*void InsertFirst(List \*L, ElType X){*

*int i;*

*for(i = Length(\*L);i>0;i--){*

*(\*L).A[i] = (\*L).A[i-1];*

*}*

*(\*L).A[0] = X;*

*}*

*/\* I.S. L terdefinisi, mungkin kosong. \*/*

*/\* F.S. v menjadi elemen pertama L. \*/*

*void InsertAt(List \*L, ElType X, IdxType i){*

*int j;*

*for(j = Length(\*L); j > i ; j--){*

*(\*L).A[j] = (\*L).A[j-1];*

*}*

*(\*L).A[i] = X;*

*}*

*/\* I.S. L terdefinisi, tidak kosong, i merupakan indeks lojik yang valid di L. \*/*

*/\* F.S. v disisipkan dalam L pada indeks ke-i (bukan menimpa elemen di i). \*/*

*void InsertLast(List \*L, ElType X){*

*(\*L).A[Length(\*L)] = X;*

*}*

*/\* I.S. L terdefinisi, mungkin kosong. \*/*

*/\* F.S. v menjadi elemen terakhir L. \*/*

*void DeleteFirst(List \*L){*

*int j;*

*for(j = 0 ; j < Length(\*L) ; j++){*

*(\*L).A[j] = (\*L).A[j+1];*

*}*

*}*

*/\* I.S. L terdefinisi, tidak kosong. \*/*

*/\* F.S. F diset dengan elemen pertama L, elemen pertama L dihapus dari L. \*/*

*void DeleteAt(List \*L, IdxType i){*

*int j;*

*for(j = i; j <Length(\*L); j++){*

*(\*L).A[j] = (\*L).A[j+1];*

*}*

*}*

*/\* I.S. L terdefinisi, tidak kosong, i merupakan indeks lojik yang valid di L. \*/*

*/\* F.S. Elemen L pada indeks ke-i dihapus dari L. \*/*

*void DeleteLast(List \*L){*

*(\*L).A[Length(\*L)-1] = Mark;*

*}*

*/\* I.S. L terdefinisi, tidak kosong. \*/*

*/\* F.S. F diset dengan elemen terakhir L, elemen terakhir L dihapus dari L. \*/*

*List Concat(List L1, List L2){*

*List L3;*

*L3 = MakeList();*

*int i,j;*

*j=0;*

*for(i = 0; i< Length(L1); i++){*

*Set(&L3,i,Get(L1,i));*

*}*

*for (i = Length(L1) ; i <(Length(L1) + Length(L2)); i++ ){*

*Set(&L3,i,Get(L2,j));*

*j++;*

*}*

*return(L3);*

*}*

*/\* Prekondisi : L1 dan L2 tidak kosong \*/*

*/\* Mengirimkan sebuah List yang merupakan gabungan dari L1 dan L2 \*/*

*/\* Urutan elemen terisi dari L1, lalu L2 \*/*

*/\* Contoh : L1 : [1, 2]; L2 : [3, 4]; Mengembalikan [1, 2, 3, 4] \*/*

**MESIN KARAKTER**

*#include <stdio.h>*

*#include "mesinkarakter.h"*

*char currentChar;*

*boolean EOP;*

*static FILE \* pita;*

*static int retval;*

*void START(){*

*pita = stdin;*

*ADV();*

*}*

*/\* Mesin siap dioperasikan. Pita disiapkan untuk dibaca.*

*Karakter pertama yang ada pada pita posisinya adalah pada jendela.*

*Pita baca diambil dari stdin.*

*I.S. : sembarang*

*F.S. : currentChar adalah karakter pertama pada pita*

*Jika currentChar != MARK maka EOP akan padam (false)*

*Jika currentChar = MARK maka EOP akan menyala (true) \*/*

*void ADV(){*

*retval = fscanf(pita,"%c",&currentChar);*

*EOP = (currentChar == MARK);*

*if(EOP){*

*fclose(pita);*

*}*

*}*

*/\* Pita dimajukan satu karakter.*

*I.S. : Karakter pada jendela = currentChar, currentChar != MARK*

*F.S. : currentChar adalah karakter berikutnya dari currentChar yang lama,*

*currentChar mungkin = MARK*

*Jika  currentChar = MARK maka EOP akan menyala (true) \*/*

*char GetCC(){*

*return(currentChar);*

*}*

*/\* Mengirimkan currentChar \*/*

*boolean IsEOP(){*

*return(currentChar == MARK);*

*}*

*/\* Mengirimkan true jika currentChar = MARK \*/*

**MESIN KATA**

*#include <stdio.h>*

*#include "mesinkata.h"*

*#include "boolean.h"*

*#include "mesinkarakterv2.h"*

*boolean EndWord;*

*Word currentWord;*

*void IgnoreBlanks(){*

*while((currentChar == BLANK) && (currentChar != MARK)){*

*ADV();*

*}*

*}*

*/\* Mengabaikan satu atau beberapa BLANK*

*I.S. : currentChar sembarang*

*F.S. : currentChar ≠ BLANK atau currentChar = MARK \*/*

*void STARTWORD(){*

*START();*

*IgnoreBlanks();*

*if(currentChar == MARK){*

*EndWord = true;*

*}*

*else {*

*EndWord = false;*

*CopyWord();*

*IgnoreBlanks();*

*}*

*}*

*/\* I.S. : currentChar sembarang*

*F.S. : EndWord = true, dan currentChar = MARK;*

*atau EndWord = false, currentWord adalah kata yang sudah diakuisisi,*

*currentChar karakter pertama sesudah karakter terakhir kata \*/*

*void ADVWORD(){*

*if(currentChar == MARK){*

*EndWord = true;*

*}*

*else{*

*CopyWord();*

*IgnoreBlanks();*

*}*

*}*

*/\* I.S. : currentChar adalah karakter pertama kata yang akan diakuisisi*

*F.S. : currentWord adalah kata terakhir yang sudah diakuisisi,*

*currentChar adalah karakter pertama dari kata berikutnya, mungkin MARK*

*Jika currentChar = MARK, EndWord = true.*

*Proses : Akuisisi kata menggunakan procedure SalinWord \*/*

*void CopyWord(){*

*int i;*

*i=0;*

*while((currentChar != MARK) && (currentChar!= BLANK) && (i < NMax)){*

*currentWord.TabWord[i] = currentChar;*

*ADV();*

*i++;*

*}*

*currentWord.Length = i;*

*}*

*/\* Mengakuisisi kata, menyimpan dalam currentWord*

*I.S. : currentChar adalah karakter pertama dari kata*

*F.S. : currentWord berisi kata yang sudah diakuisisi;*

*currentChar = BLANK atau currentChar = MARK;*

*currentChar adalah karakter sesudah karakter terakhir yang diakuisisi.*

*Jika panjang kata melebihi NMax, maka sisa kata "dipotong" \*/*

**POINT**

*#include<stdio.h>*

*#include<math.h>*

*#include "point.h"*

*#include "boolean.h"*

*POINT MakePOINT (float X, float Y){*

*POINT pt;*

*Absis(pt) = X;*

*Ordinat(pt) = Y;*

*return pt;*

*}*

*void BacaPOINT (POINT \* P){*

*float x,y;*

*scanf("%f %f",&x,&y);*

*Absis(\*P) = x;*

*Ordinat(\*P) = y;*

*}*

*void TulisPOINT (POINT P){*

*printf("(%.2f,%.2f)",Absis(P),Ordinat(P));*

*}*

*boolean EQ (POINT P1, POINT P2){*

*if((Absis(P1) == Absis(P2)) && (Ordinat(P1)==Ordinat(P2))){*

*return true;*

*}*

*else{*

*return false;*

*}*

*}*

*boolean NEQ (POINT P1, POINT P2){*

*if((Absis(P1) != Absis(P2)) || (Ordinat(P1)!=Ordinat(P2))){*

*return true;*

*}*

*else{*

*return false;*

*}*

*}*

*boolean IsOrigin (POINT P){*

*if((Absis(P) == 0) && (Ordinat(P)== 0)){*

*return true;*

*}*

*else{*

*return false;*

*}*

*}*

*boolean IsOnSbX (POINT P){*

*if((Ordinat(P)== 0)){*

*return true;*

*}*

*else{*

*return false;*

*}*

*}*

*boolean IsOnSbY (POINT P){*

*if((Absis(P) == 0)){*

*return true;*

*}*

*else{*

*return false;*

*}*

*}*

*int Kuadran (POINT P){*

*int kuadran;*

*if((Absis(P)>0) && (Ordinat(P)>0)){*

*kuadran = 1;*

*}*

*else if((Absis(P)>0) && (Ordinat(P)<0)){*

*kuadran = 4;*

*}*

*else if((Absis(P)<0) && (Ordinat(P)>0)){*

*kuadran = 2;*

*}*

*else if((Absis(P)<0) && (Ordinat(P)<0)){*

*kuadran = 3;*

*}*

*return kuadran;*

*}*

*POINT PlusDelta (POINT P, float deltaX, float deltaY){*

*Absis(P) = Absis(P) + deltaX;*

*Ordinat(P) = Ordinat(P) + deltaY;*

*return P;*

*}*

*void Geser (POINT \*P, float deltaX, float deltaY){*

*Absis(\*P) = Absis(\*P) + deltaX;*

*Ordinat(\*P) = Ordinat(\*P) + deltaY;*

*}*

*float Jarak0 (POINT P){*

*float tegak, datar, miring;*

*datar = powf(Absis(P),2);*

*tegak = powf(Ordinat(P),2);*

*miring = sqrtf(datar + tegak);*

*return miring;*

*}*

*float Panjang (POINT P1, POINT P2){*

*float dx,dy,hasil;*

*dx = Absis(P1) - Absis(P2);*

*dy = Ordinat(P1) - Ordinat(P2);*

*dx = powf(dx,2);*

*dy = powf(dy,2);*

*hasil = sqrtf(dx + dy);*

*return hasil;*

*}*

**UNTITLED**

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