*#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] \*/*