STRUCTURES & UNIONS

Soal no 1:

#include <stdio.h>

struct Address {

    char street\_name[40];

    int number;

    char city[20];

    char province[20];

};

struct Dob{

    int date, month, year;

};

struct Student{

    int student\_number;

    char name[20];

    struct Address addr;

    char place[20];

    struct Dob dateofbirth;

};

int main (){

    return 0;

}

Soal no 2:

#include <stdio.h>

struct Address {

    char street\_name[40];

    int number;

    char city[20];

    char province[20];

};

struct Dob{

    int date, month, year;

};

struct Student{

    int student\_number;

    char name[20];

    struct Address addr;

    char place[20];

    struct Dob dateofbirth;

};

int main (){

    struct Student daftar[5];

    for (int i = 0; i < 1; i++)

    {

        printf("insert student number : \n");

        scanf("%d",&daftar[i].student\_number);

        printf("insert student name : \n");

        scanf("%s",&daftar[i].name);

        printf("Insert student address(street name, number, city, province) : \n");

        scanf("%s %d %s %s",&daftar[i].addr.street\_name,&daftar[i].addr.number,&daftar[i].addr.city,&daftar[i].addr.province);

        getchar();

        printf("Insert student place : \n");

        scanf("%s",&daftar[i].place);

        printf("Insert Student date of birth(date, month, and year) : \n");

        scanf("%d %d %d",&daftar[i].dateofbirth.date,&daftar[i].dateofbirth.month,&daftar[i].dateofbirth.year);

    }

    // optional

    // for (int j = 0; j < 5; j++)

    // {

    //     printf("%d\n",daftar[j].student\_number);

    //     printf("%s\n",daftar[j].name);

    //     printf("%s %d %s %s\n",daftar[j].addr.street\_name,daftar[j].addr.number,daftar[j].addr.city,daftar[j].addr.province);

    //     printf("%s\n",daftar[j].place);

    //     printf("%d %d %d\n",daftar[j].dateofbirth.date,daftar[j].dateofbirth.month,daftar[j].dateofbirth.year);

    // }

    return 0;

}

MEMORY ALLOCATION

Soal no 1:

#include <stdio.h>

int  main(){

    // void \*malloc ( size\_t size );

    // - malloc() = is used to allocate one block of memory dynamically at heap memory.

    //   stating the size of allocated memory (in byte)

    // -  ( size\_t size ) = size of memory in byte.

    // void \*calloc ( size\_t num, size\_t size );

    // - calloc() = is used to allocate some blocks of memory dynamically at heap memory,

    //   where each element given size.

    // - size\_t num = number of byte each block or element

    // - size\_t size  = number of byte each block or element

    // void \*realloc ( void \*memblock, size\_t size );

    // - realloc() is used to reallocate some blocks of memory or element dynamically at heap.

    // - void \*memblock = is a pointer to the address of targeted memory location

    // - size\_t size  = number of byte each block or element

    // void free ( void \*memblock );

    // - free() =  used to release allocated memory at heap memory.

    // - void \*memblock = is a pointer to the address of targeted memory location

    return 0;

}

Soal no 2

#include <stdio.h>

int main (){

// Pointer to function is the address of a function in the memory

// we can use pointers to call functions and to pass functions as arguments to other functions

// void (\*ptr[3])(int)= {function1. function2, function3};

// Describe the above code!

// to assign in pointer ptr[3](array's adress) to a function returning an int

// to assign function1. function2, function3 to "pointer to functions"

// mendaftarkan function1, function2, function3,

// ke alamat arrays yang ditunjuk oleh pointer yakni \*ptr[3]

    return 0;

}