**LATIHAN IMPLEMENTASI TREE**

KELOMPOK 4:

- DANGIANG RAKEAN A (2106902)

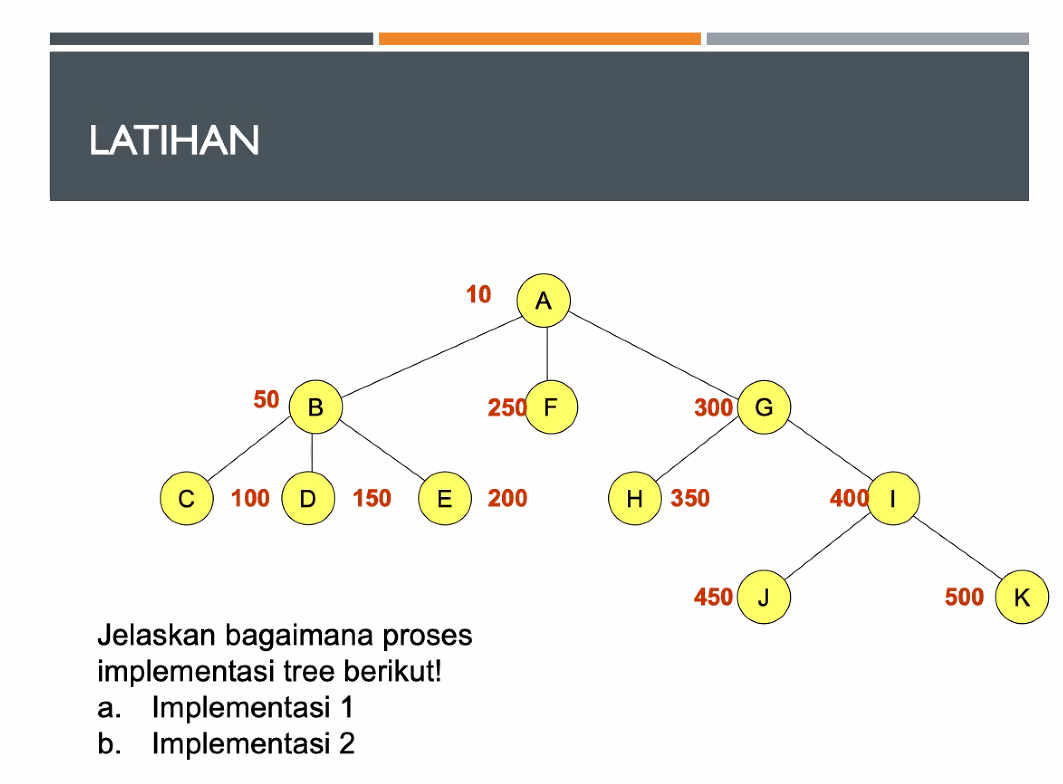
- KHAIRUNNISA (2103683)

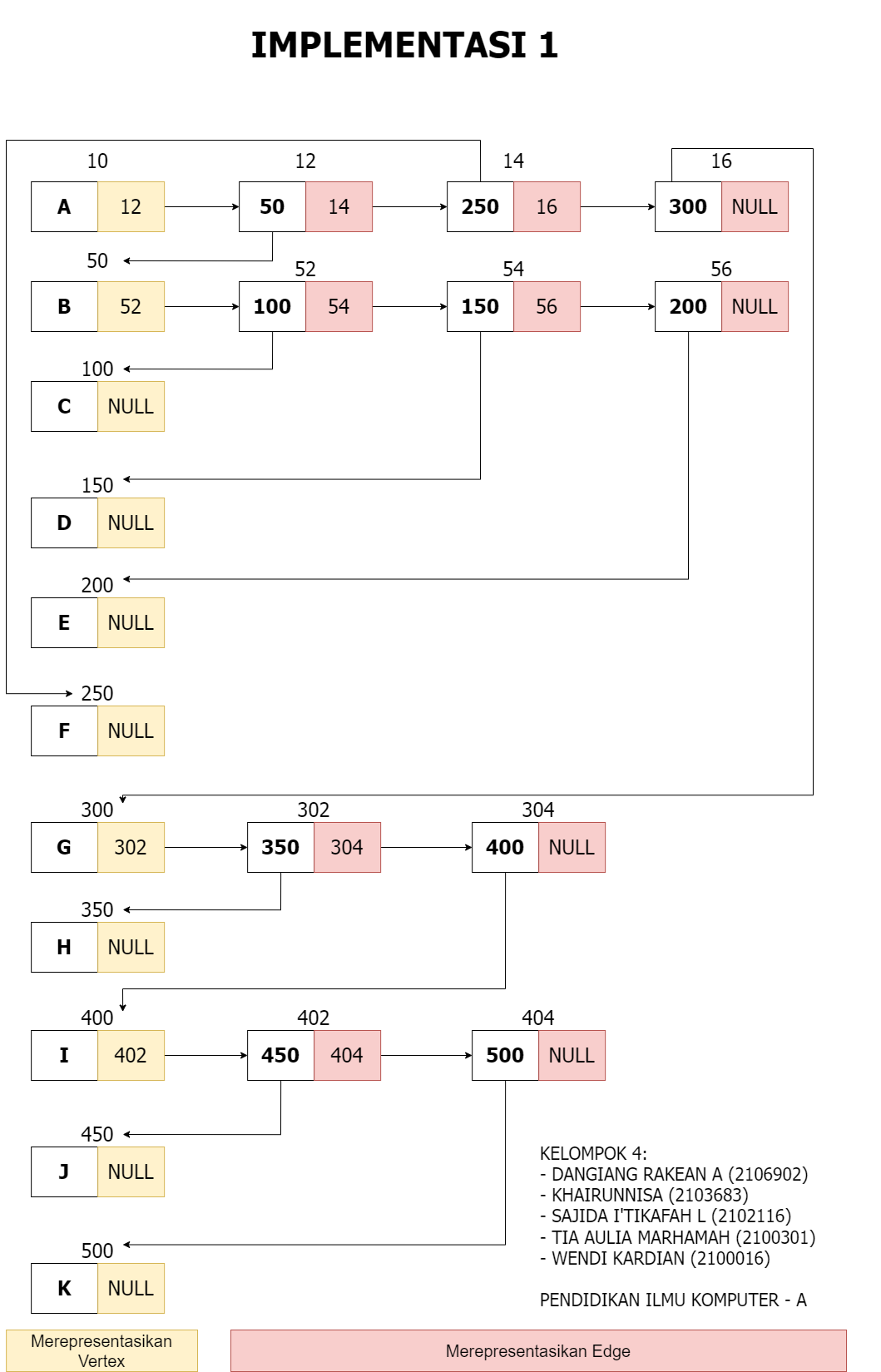
- SAJIDA I'TIKAFAH L (2102116)

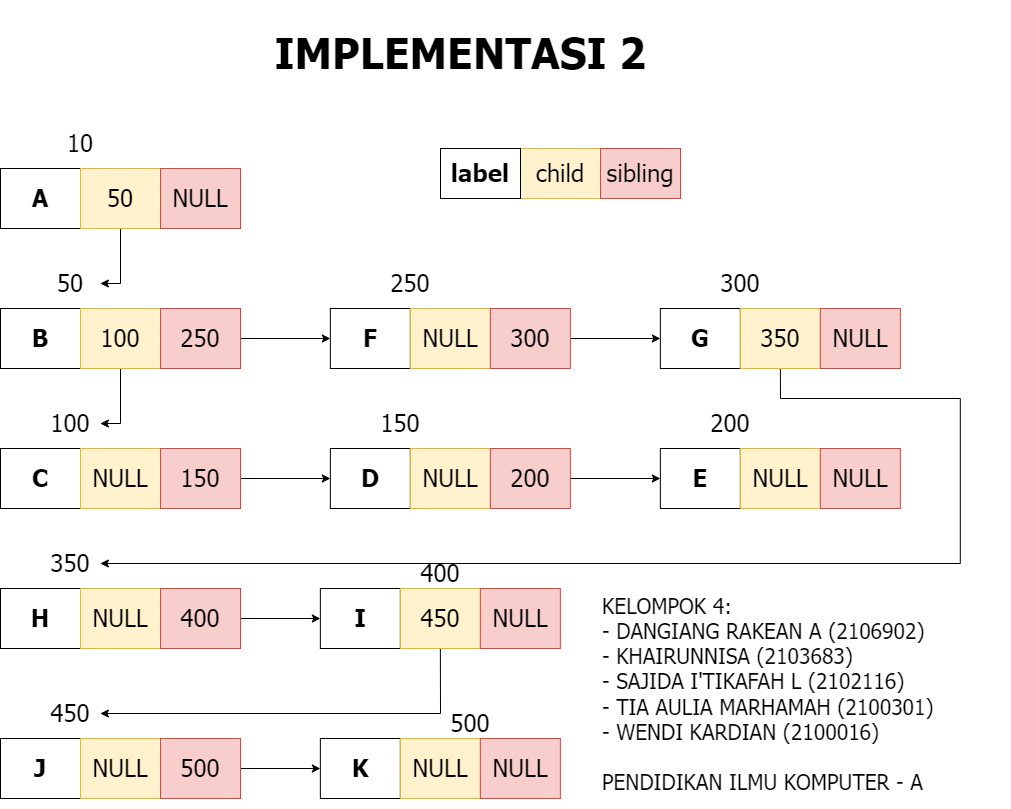
- TIA AULIA MARHAMAH (2100301)

- WENDI KARDIAN (2100016)

PENDIDIKAN ILMU KOMPUTER – A







**Source Code Implementasi 1**

Untuk lebih jelas dapat kunjungi link Github : [Data-Structure-using-C/implementasi1.c at master · wendikardian/Data-Structure-using-C (github.com)](https://github.com/wendikardian/Data-Structure-using-C/blob/master/Pertemuan%2012/implementasi1.c)

// STRUKTUR DATA TREE STATIC

// WENDI KARDIAN (2100016) - Pendidikan Ilmu Komputer - A

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\*struct untuk vertex\*/

typedef struct simpul{

    struct ruas \*jalur;

    struct simpul \*nextVertex;

    char label;

    char id[2];

}simpul;

/\*struct untuk edge\*/

typedef struct ruas{

    struct ruas \*nextEdge;

    struct simpul \*vertexTujuan;

    char bobot[2];

}ruas;

simpul \*awal = NULL;

// Membuat vertex baru

simpul \*createVertex (char a, char b[]){

    simpul \*simpulBaru = (simpul\*)malloc(sizeof(simpul));

    simpulBaru->label = a;

    strcpy(simpulBaru->id,b);

    simpulBaru->jalur = NULL;

    simpulBaru->nextVertex = NULL;

    return simpulBaru;

}

/\*fungsi untuk menemukan vertex didalam graph\*/

simpul \*cariSimpul(char a) {

    simpul \*bantu = awal;

    if(bantu != NULL){

        while(bantu->nextVertex != NULL){

            if(bantu->label == a){

                break;

            }

            bantu = bantu->nextVertex;

        }

    }

    return bantu;

}

void tambahVertex(char a, char b[]){ //menanbahkan vertex berdasarkan label dan idnya lalu disambungkan

    simpul \*prev = cariSimpul(a);

    if(prev == NULL){

        simpul \*baru = createVertex(a, b);

        awal = baru;

    }

    else{

        if((prev->nextVertex == NULL) && (prev->label != a)){

            simpul \*baru = createVertex(a, b);

            prev->nextVertex = baru;

        }

    }

}

void createEdge (simpul \*a, simpul \*t, char bobot[]){ //membuat hubungan antar vertex

    ruas \*newEdge = (ruas\*)malloc(sizeof(ruas));

    strcpy(newEdge->bobot,bobot);

    newEdge->nextEdge = NULL;

    newEdge->vertexTujuan = t;

    if (a->jalur == NULL){

        a->jalur = newEdge;

    }

    else{

        ruas \*jalurAkhir = a->jalur;

        while (jalurAkhir->nextEdge != NULL){

            jalurAkhir = jalurAkhir->nextEdge;

        }

        jalurAkhir->nextEdge = newEdge;

    }

}

void tambahEdge(char Vasal, char nilaiEdge[], char Vtujuan){ //menambahkan edge

    simpul \*a,\*t;

    a = cariSimpul(Vasal);

    t = cariSimpul(Vtujuan);

    createEdge(a,t,nilaiEdge);

}

// Display the graph

void display(){

    simpul \*tempSimpul = awal;

    printf("|-----------------------------------------------------------| \n");

    printf("|                 NILAI TREE                                | \n");

    printf("|-----------------------------------------------------------| \n");

    if (tempSimpul != NULL) {

        while (tempSimpul != NULL){

            if(tempSimpul->jalur != NULL){

                printf("\n Vertex %c yang memiliki id %s Memiliki Child : \n", tempSimpul->label, tempSimpul->id);

                ruas \*tempEdge = tempSimpul->jalur;

                while (tempEdge != NULL){

                    printf(" ------ vertex %c \n",tempEdge->vertexTujuan->label);

                    tempEdge = tempEdge->nextEdge;

                }

                printf("-----------------------------------------------------------\n");

                printf("\n");

                tempSimpul = tempSimpul->nextVertex;

            }else{

                tempSimpul = tempSimpul->nextVertex;

            }

        }

    }

    else{

        printf("Graph Kosong");

    }

}

void checkVertex(){

    simpul \*tempSimpul = awal;

    while(tempSimpul != NULL){

        printf("%c\n", tempSimpul->label);

        tempSimpul = tempSimpul->nextVertex;

    }

}

int main(){

    tambahVertex('A', "v1");

    tambahVertex('B', "v2");

    tambahVertex('C', "v3");

    tambahVertex('D', "v4");

    tambahVertex('E', "v5");

    tambahVertex('E', "v5");

    tambahVertex('F', "v6");

    tambahVertex('G', "v7");

    tambahVertex('H', "v8");

    tambahVertex('I', "v9");

    tambahVertex('J', "v10");

    tambahVertex('K', "v11");

    tambahEdge('A',"5",'B');

    tambahEdge('A',"e2",'F');

    tambahEdge('A',"e2",'G');

    tambahEdge('B',"e2",'C');

    tambahEdge('B',"e3",'D');

    tambahEdge('B',"e4",'E');

    tambahEdge('G',"4",'H');

    tambahEdge('G',"7",'I');

    tambahEdge('I',"3",'J');

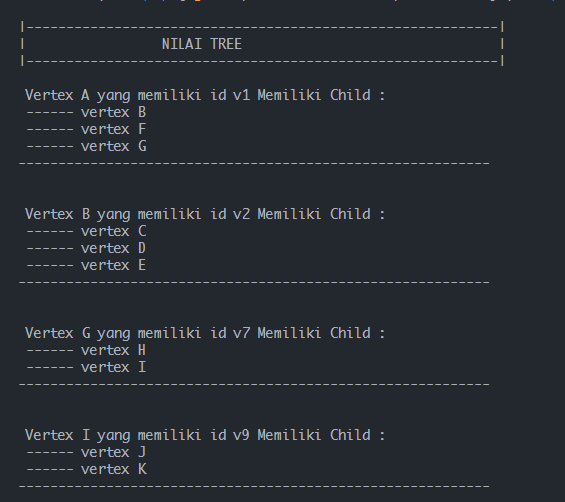
    tambahEdge('I',"9",'K');

    display();

    return 0;

}

Output SC Implementasi 1 :



**Source Code Implementasi 2**

Untuk lebih jelas dapat kunjungi link Github : [Data-Structure-using-C/implementasi2.c at master · wendikardian/Data-Structure-using-C (github.com)](https://github.com/wendikardian/Data-Structure-using-C/blob/master/Pertemuan%2012/implementasi2.c)