TUGAS PENDAHULUAN 4

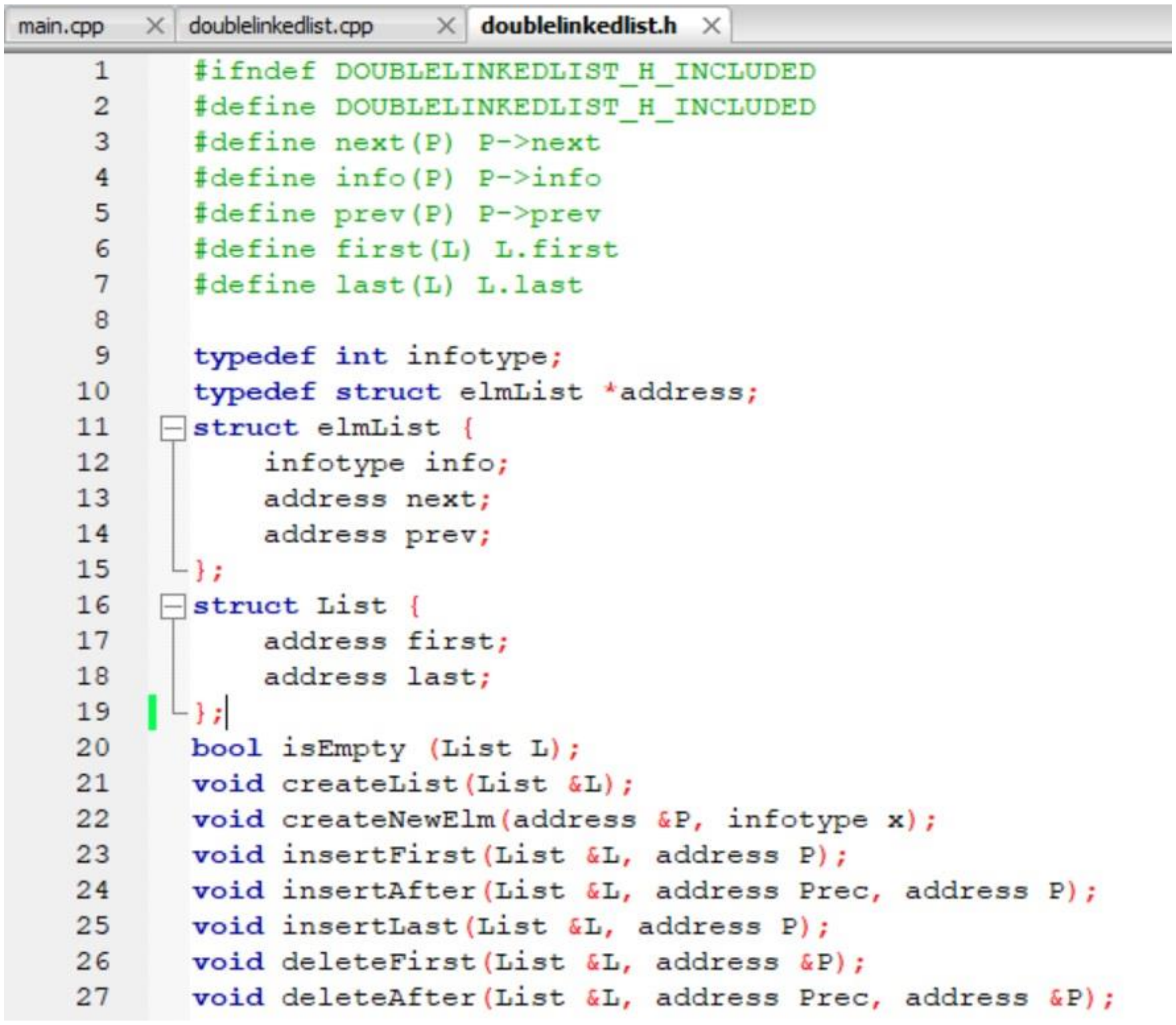
STRUKTUR DATA

Nama : Bagas Tri Wibowo

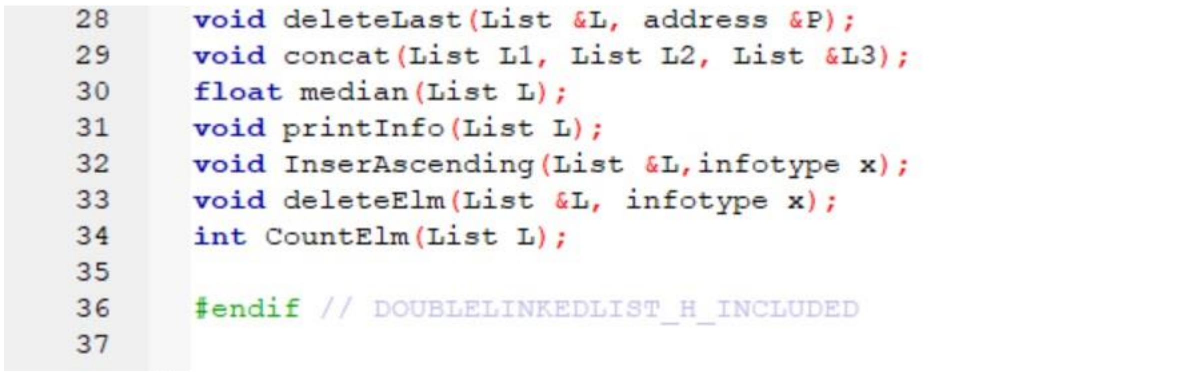
NIM : 1301194051

Kelas : IF-43-04

Header :

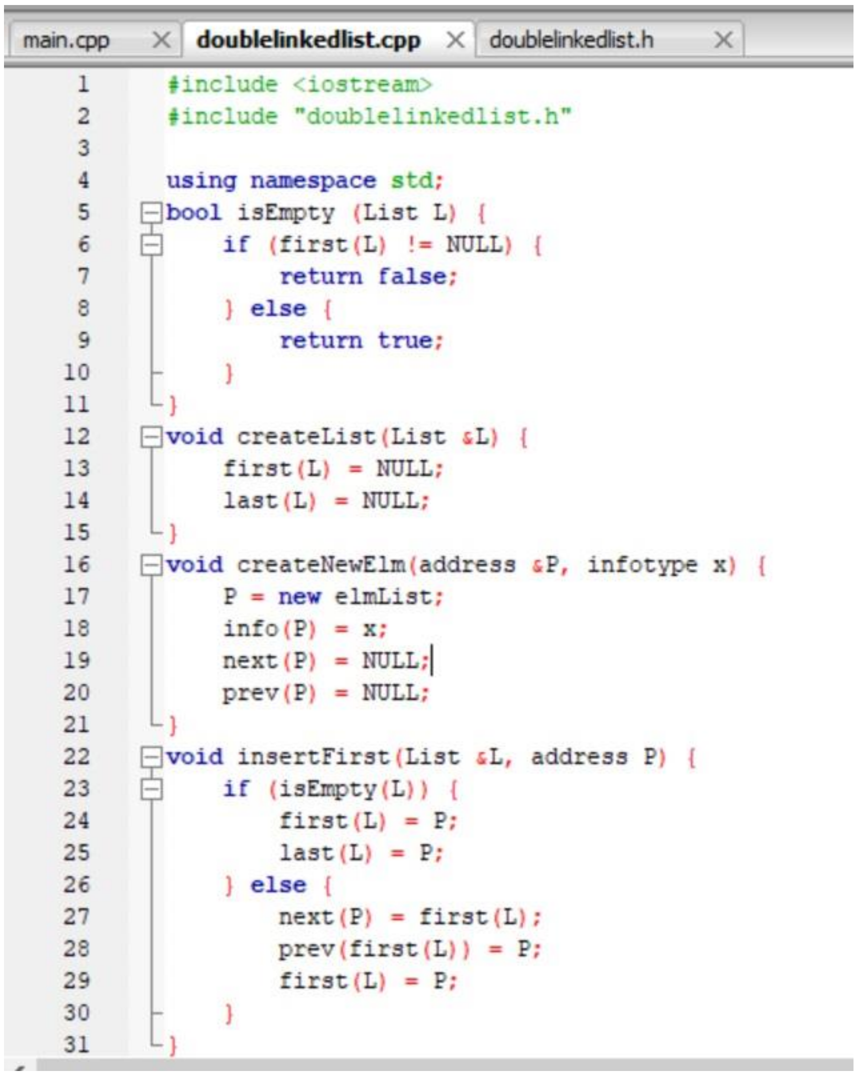


```cpp
#ifndef DOUBLELINKEDLIST_H_INCLUDED
#define DOUBLELINKEDLIST_H_INCLUDED
#define next(P) P->next
#define info(P) P->info
#define prev(P) P->prev
#define first(L) L.first
#define last(L) L.last

typedef int infotype;
typedef struct elmList *address;
struct elmList {
    infotype info;
    address next;
    address prev;
};
struct List {
    address first;
    address last;
};
bool isEmpty (List L);
void createList(List &L);
void createNewElm(address &P, infotype x);
void insertFirst(List &L, address P);
void insertAfter(List &L, address Prec, address P);
void insertLast(List &L, address P);
void deleteFirst(List &L, address &P);
void deleteAfter(List &L, address Prec, address &P);
```



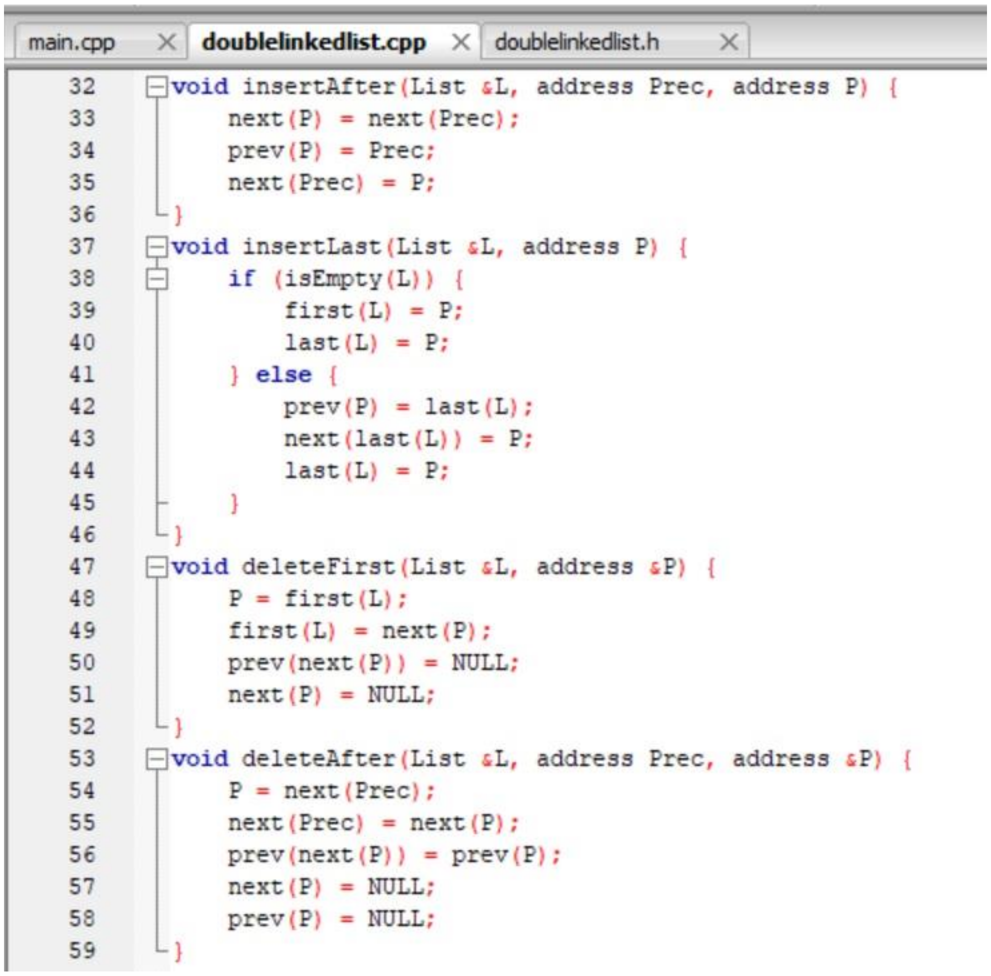
```cpp
void deleteLast(List &L, address &P);
void concat(List L1, List L2, List &L3);
float median(List L);
void printInfo(List L);
void InserAscending(List &L,infotype x);
void deleteElm(List &L, infotype x);
int CountElm(List L);

#endif // DOUBLELINKEDLIST_H_INCLUDED

```

Implementation :

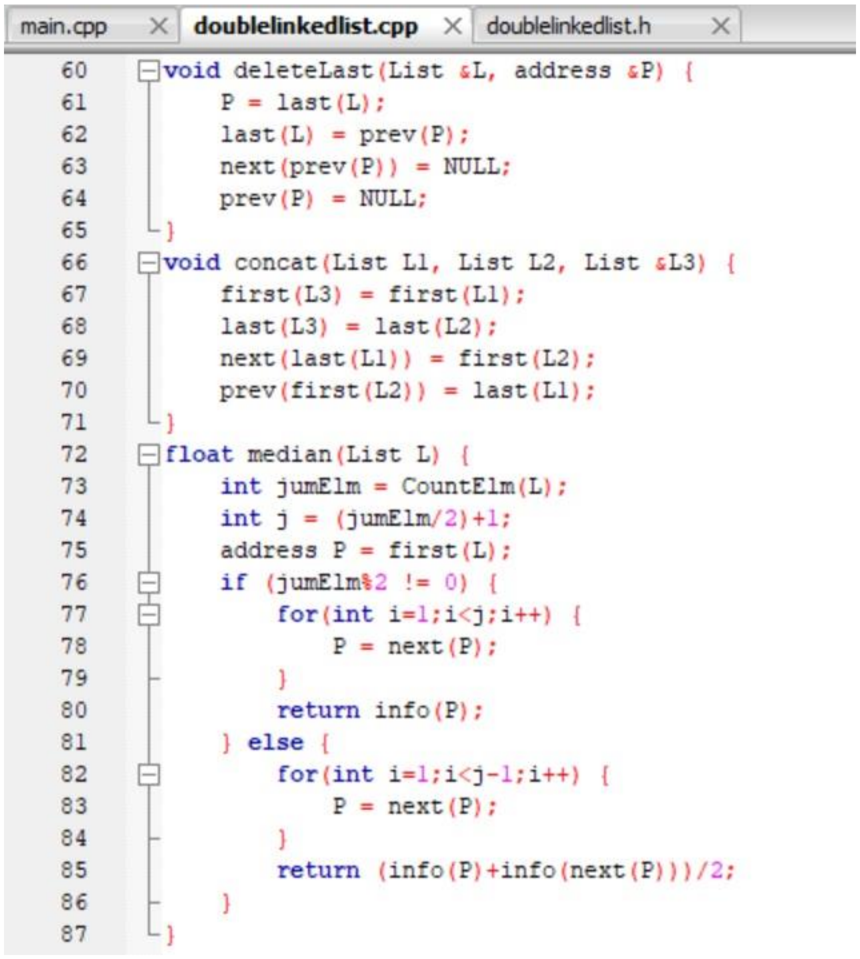


main.cpp doublelinkedlist.cpp doublelinkedlist.h

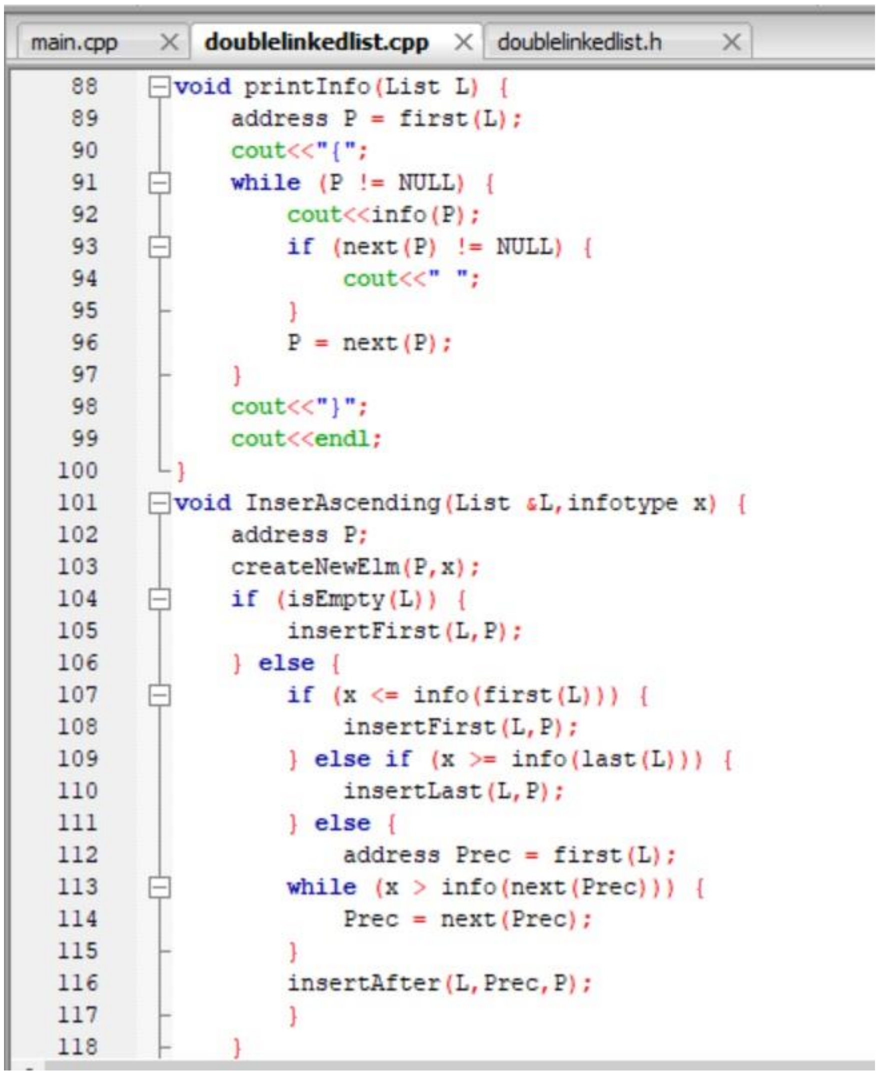
```cpp
#include <iostream>
#include "doublelinkedlist.h"

using namespace std;
bool isEmpty (List L) {
    if (first(L) != NULL) {
        return false;
    } else {
        return true;
    }
}
void createList(List &L) {
    first(L) = NULL;
    last(L) = NULL;
}
void createNewElm(address &P, infotype x) {
    P = new elmList;
    info(P) = x;
    next(P) = NULL;
    prev(P) = NULL;
}
void insertFirst(List &L, address P) {
    if (isEmpty(L)) {
        first(L) = P;
        last(L) = P;
    } else {
        next(P) = first(L);
        prev(first(L)) = P;
        first(L) = P;
    }
}
```



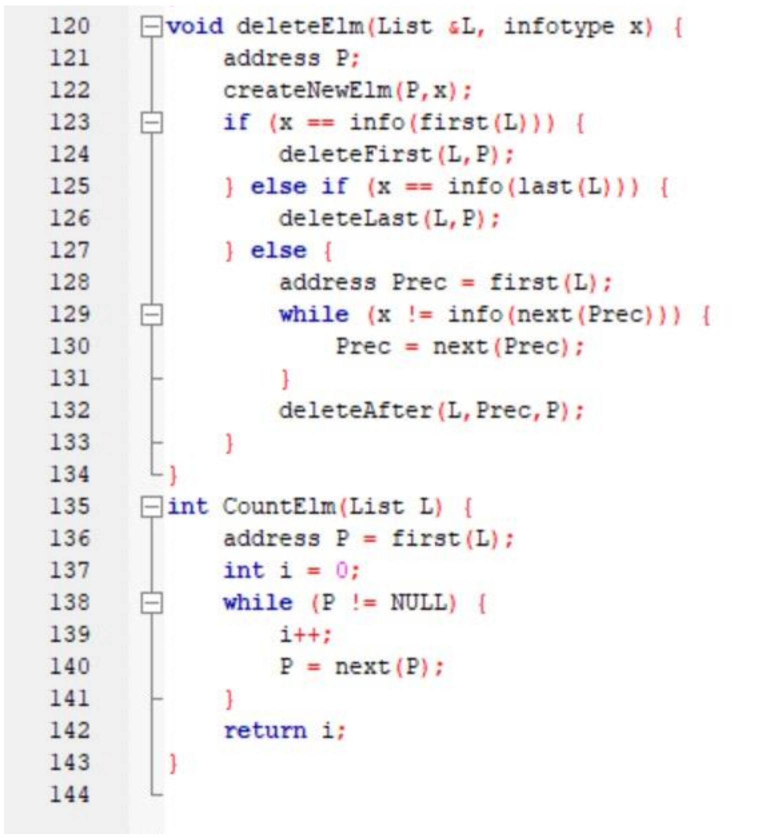
```
void insertAfter(List &L, address Prec, address P) {
    next(P) = next(Prec);
    prev(P) = Prec;
    next(Prec) = P;
}
void insertLast(List &L, address P) {
    if (isEmpty(L)) {
        first(L) = P;
        last(L) = P;
    } else {
        prev(P) = last(L);
        next(last(L)) = P;
        last(L) = P;
    }
}
void deleteFirst(List &L, address &P) {
    P = first(L);
    first(L) = next(P);
    prev(next(P)) = NULL;
    next(P) = NULL;
}
void deleteAfter(List &L, address Prec, address &P) {
    P = next(Prec);
    next(Prec) = next(P);
    prev(next(P)) = prev(P);
    next(P) = NULL;
    prev(P) = NULL;
}
```



```
void deleteLast(List &L, address &P) {
    P = last(L);
    last(L) = prev(P);
    next(prev(P)) = NULL;
    prev(P) = NULL;
}
void concat(List L1, List L2, List &L3) {
    first(L3) = first(L1);
    last(L3) = last(L2);
    next(last(L1)) = first(L2);
    prev(first(L2)) = last(L1);
}
float median(List L) {
    int jumElm = CountElm(L);
    int j = (jumElm/2)+1;
    address P = first(L);
    if (jumElm%2 != 0) {
        for(int i=1;i<j;i++) {
            P = next(P);
        }
        return info(P);
    } else {
        for(int i=1;i<j-1;i++) {
            P = next(P);
        }
        return (info(P)+info(next(P)))/2;
    }
}
```

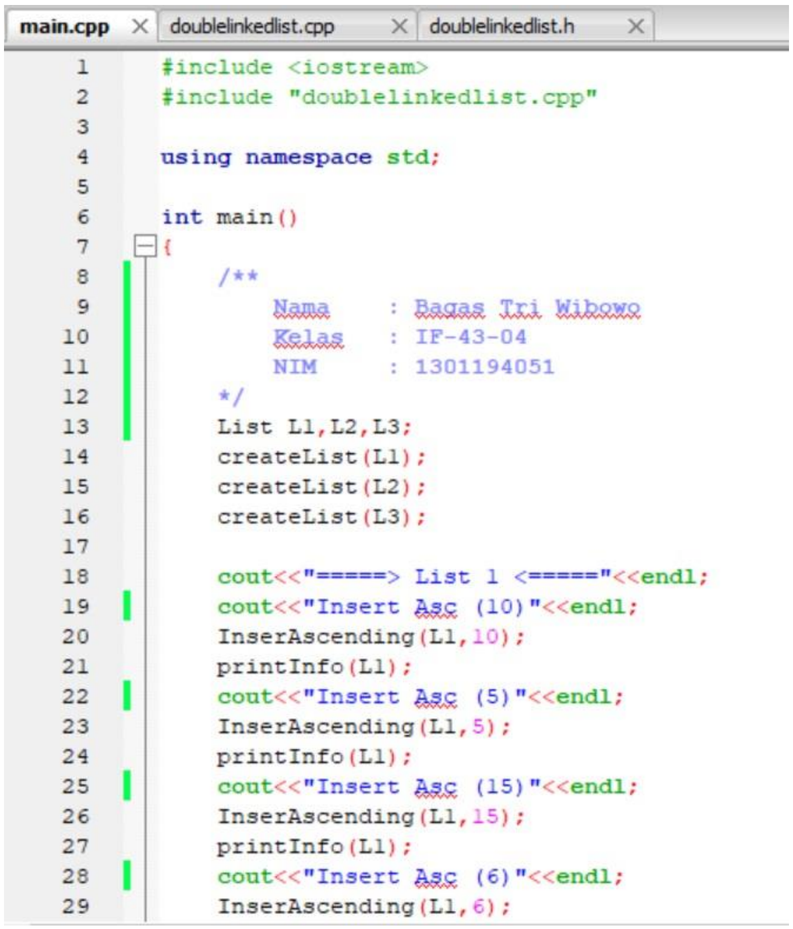


```cpp
void printInfo(List L) {
    address P = first(L);
    cout<<"{";
    while (P != NULL) {
        cout<<info(P);
        if (next(P) != NULL) {
            cout<<" ";
        }
        P = next(P);
    }
    cout<<"}";
    cout<<endl;
}
void InserAscending(List &L,infotype x) {
    address P;
    createNewElm(P,x);
    if (isEmpty(L)) {
        insertFirst(L,P);
    } else {
        if (x <= info(first(L))) {
            insertFirst(L,P);
        } else if (x >= info(last(L))) {
            insertLast(L,P);
        } else {
            address Prec = first(L);
        while (x > info(next(Prec))) {
            Prec = next(Prec);
        }
        insertAfter(L,Prec,P);
        }
    }
```

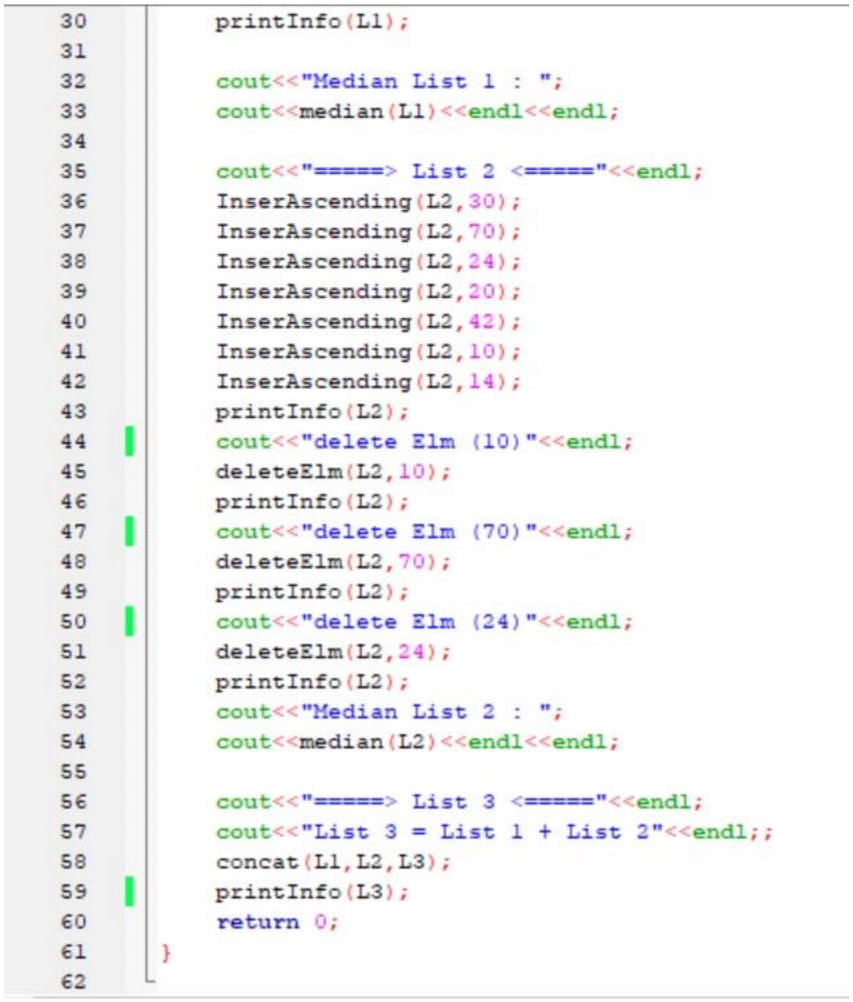


```cpp
void deleteElm(List &L, infotype x) {
    address P;
    createNewElm(P,x);
    if (x == info(first(L))) {
        deleteFirst(L,P);
    } else if (x == info(last(L))) {
        deleteLast(L,P);
    } else {
        address Prec = first(L);
        while (x != info(next(Prec))) {
            Prec = next(Prec);
        }
        deleteAfter(L,Prec,P);
    }
}
int CountElm(List L) {
    address P = first(L);
    int i = 0;
    while (P != NULL) {
        i++;
        P = next(P);
    }
    return i;
}
```

Main Program :

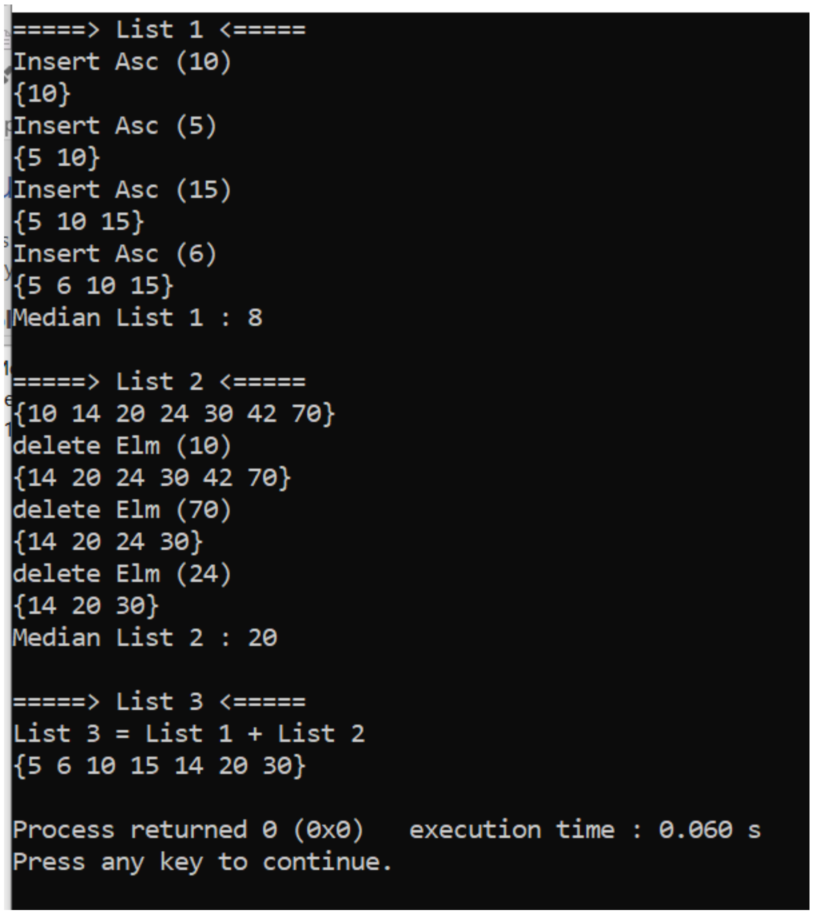


```
#include <iostream>
#include "doublelinkedlist.cpp"

using namespace std;

int main()
{
    /**
        Nama     : Bagas Tri Wibowo
        Kelas    : IF-43-04
        NIM      : 1301194051
    */
    List L1,L2,L3;
    createList(L1);
    createList(L2);
    createList(L3);

    cout<<"=====> List 1 <====="<<endl;
    cout<<"Insert Asc (10)"<<endl;
    InserAscending(L1,10);
    printInfo(L1);
    cout<<"Insert Asc (5)"<<endl;
    InserAscending(L1,5);
    printInfo(L1);
    cout<<"Insert Asc (15)"<<endl;
    InserAscending(L1,15);
    printInfo(L1);
    cout<<"Insert Asc (6)"<<endl;
    InserAscending(L1,6);
```



```
    printInfo(L1);

    cout<<"Median List 1 : ";
    cout<<median(L1)<<endl<<endl;

    cout<<"=====> List 2 <====="<<endl;
    InserAscending(L2,30);
    InserAscending(L2,70);
    InserAscending(L2,24);
    InserAscending(L2,20);
    InserAscending(L2,42);
    InserAscending(L2,10);
    InserAscending(L2,14);
    printInfo(L2);
    cout<<"delete Elm (10)"<<endl;
    deleteElm(L2,10);
    printInfo(L2);
    cout<<"delete Elm (70)"<<endl;
    deleteElm(L2,70);
    printInfo(L2);
    cout<<"delete Elm (24)"<<endl;
    deleteElm(L2,24);
    printInfo(L2);
    cout<<"Median List 2 : ";
    cout<<median(L2)<<endl<<endl;

    cout<<"=====> List 3 <====="<<endl;
    cout<<"List 3 = List 1 + List 2"<<endl;;
    concat(L1,L2,L3);
    printInfo(L3);
    return 0;
}
```

Output :



```
=====> List 1 <=====
Insert Asc (10)
{10}
Insert Asc (5)
{5 10}
Insert Asc (15)
{5 10 15}
Insert Asc (6)
{5 6 10 15}
Median List 1 : 8

=====> List 2 <=====
{10 14 20 24 30 42 70}
delete Elm (10)
{14 20 24 30 42 70}
delete Elm (70)
{14 20 24 30}
delete Elm (24)
{14 20 30}
Median List 2 : 20

=====> List 3 <=====
List 3 = List 1 + List 2
{5 6 10 15 14 20 30}

Process returned 0 (0x0)   execution time : 0.060 s
Press any key to continue.
```