var_list.c

```c
#include<stdio.h>
#include <stddef.h>
#include<string.h>
#include<stdlib.h>
#include "var_list.h"
#include <stdbool.h>

const int KEYS_SIZE = 25;
char KEYS[50][25] = {
    "void", "int","double","float","justInCase",
    "println","discard","till","import",
    "static","void","entryPoint",
    "lt","gt","eq","neq","le","ge",
    "add", "sub", "mul", "div", "dif", "rem"
};

struct VARIABLE *head = NULL;
struct VARIABLE *tail = NULL;

struct VARIABLE* createNode(const char *name, char *type, double value) {
    struct VARIABLE *newNode = (struct VARIABLE*) malloc(sizeof(struct VARIABLE));
    if(!newNode) {
        printf("Memory allocation failed.\n");
        return NULL;
    }

    strncpy(newNode->name, name, sizeof(newNode->name) - 1);
    newNode->value = value;
    strncpy(newNode->type,type,sizeof(newNode->type)-1);
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertVariable(char *name, char *type, double val) {

    for(int i=0; i<KEYS_SIZE; i++){
        if(strcmp(KEYS[i],name) == 0){
            printf("Keyword '%s' can't be variable\n",name);
            return;
        }
    }

    struct VARIABLE *var = createNode(name,type,val);

    if( strncmp("int",type,3) == 0 ){ // integer
        val = (double)( (int)val ); // ignoring after decimal
    }

    if(tail == NULL) {
        head = var;
        tail = var;
    }
    else {
        var->prev = tail;
        tail->next = var;
        tail = var;
    }
}

int getTotalVar(){
    struct VARIABLE *ptr;
    int count=0;
    ptr = head;
    while(ptr != NULL){
        count++;
        ptr = ptr->next;
    }
    return count;
}

void updateVariable(char *name, double val){

    struct VARIABLE *ptr;
    ptr = head;
    while(ptr != NULL){
        if( strcmp(ptr->name,name) == 0 ){

            if( strncmp("int",ptr->type,3) == 0 ) val = (int)val;

            ptr->value = val;
            break;
        }
        ptr = ptr->next;
    }
}

void deleteVariable(char *name){
    struct VARIABLE *ptr;
    ptr = head;
    while(ptr != NULL){
        if( strcmp(ptr->name, name) == 0 ){
            // first delete
            if(ptr == head){
                //single node
                if(ptr == tail) tail = NULL;
                head = ptr->next;
            }
            else if(ptr == tail){ // last delete
                tail = tail->prev;
                tail->next = NULL;
            }
            else{
                ptr->prev->next = ptr->next;
                ptr->next->prev = ptr->prev;
            }
        }
        ptr = ptr->next;
    }
```

```c
}

bool doesVariableExists(char *name){
    struct VARIABLE *ptr = head;

    while (ptr != NULL)
    {
        if( strcmp(name,ptr->name) == 0 ) return true;
        ptr = ptr->next;
    }
    return false;
}

struct VARIABLE* getVariable(char* name){
    //printf("printing from inner\n");
    //printAll();
    struct VARIABLE *ptr = head;
    while (ptr != NULL)
    {
        //printf("(%s %ld),",ptr->name,ptr->value);
        if( strcmp(name,ptr->name) == 0 ){
            //printf("\n value returning %s %lf\n",ptr->name,ptr->value);
            return ptr;
        }
        ptr = ptr->next;
    }
    return NULL;
}

double getValueOrDefault(char* name){
    struct VARIABLE *ptr = head;
    while (ptr != NULL)
    {
        if( strcmp(name,ptr->name) == 0 ){
            return ptr->value;
        }
        ptr = ptr->next;
    }
    return 0;
}

char* getFormattedValueOrDefault(char *name){
    struct VARIABLE* var = getVariable(name);
    if(var == NULL){
        return "0";
    }

    char *arr = (char *) malloc(20);

    if ( strcmp(var->type,"int") == 0){
        int num = (int)(var->value);
        sprintf(arr, "%d", num);
        return arr;
    }

    double num = (var->value);
    sprintf(arr, "%lf", num);
    return arr;
}

void printAll(){
    printf("\n");
    if(head == NULL) return;


    printf("----------Printing all variables----------\n");

    struct VARIABLE *ptr;
    ptr = head;
    while(ptr != NULL){
        if ( strcmp(ptr->type,"int") == 0){
            printf("%s(%s) %d -> ",ptr->name,ptr->type,(int)ptr->value);
        }
        else{
            printf("%s(%s) %lf -> ",ptr->name,ptr->type,ptr->value);
        }
        ptr = ptr->next;
    }
    printf("\n\n");
}


bool *validityList = NULL;
int validIndex = 0;

void pushValidity(bool val){
    validIndex++;
    validityList = (bool*) realloc(validityList,validIndex);
    validityList[validIndex-1] = val;
}

bool getCurrentValidity(){
    if(validIndex <= 0) return true;
    return validityList[validIndex-1];
}

bool popValidity(){
    bool val = validityList[validIndex];
    validIndex--;
    return val;
}

```