****
Problem Statement 1
****

Submission By :
- Ameesha Mittal (2014A7PS107P)
- M Sharat Chandra (2014A7PS108P)
- Vidhi Jain (2014A7PS113P)

Main functors:
****

is_correct([[[A,B]|T],L]).

This is the main function which takes a single input as list.
Here [[A,B]|T] represents a list consisting of [A,B] type sub lists.
- A is the variable, for example: x,y,z
- B is the datatype of the variable: 'int','float','boolean'
- L is the expression to be type-checked (in infix notation).

is_correct([[[A,B,C]|T],L]).

This form of is_correct will be called when address variable needs to be initialised.
Here [[A,B,C]|T] represents a list consisting of [A,B,C] type sub lists.
- A is datatype 'address'.
- B is datatype of the variable ,e.g. 'int','float','boolean','bitset'.
- C is the variable name ,e.g. x,y,z

examples:

is_correct([[[x,'int'],[y,'int']],[[2,'<',3],'&&',[x,'<',y]]]).

is_correct([[['address','int','x'],['address','int','y']],['*','x','+','*','y']]).

is_correct_no_var(L).

- If we don't want to initialise any variable, we can directly input the test expression (L) in this function.

NOTE:
****

- All the operators must be enclosed in single quotes.
- Numbers are not to be enclosed in single quotes.
- Each term in the expression list should be separated by comma.
- Enclose the sub-expressions in square brackets to set precedence wherever required.
- MUST enclose in square brackets:
    * Expression on both sides of '='
    * Expression containing comparators
    * Conditional Expressions in '?:' operator

Sub functors:
****

integer_expr()   : For evaluating integer expressions (arithmetic and address subtraction)

float_expr()   : For evaluating float expressions

boolean_expr()   : For evaluating boolean expressions

bitset_expr()   : For evaluating bitset expressions

address_expr()   : For evaluating expression involving address and pointers

equals_expr([A,B,C])   : For evaluating assignment operation. Here A and C should be of same type.

conditional_expr([E1,E2,E3,E4,E5])   : For evaluating conditional expressions (?:).Here E1 should be boolean. E3 and E5 should be of same type.

type_assign(X,Y)   : For assigning the variable X to data type Y

replace(X,[A,B],Z)   : X is a list in which replacement is to be made. Whenever in list X we encounter A, we will replace it with B. Z is the list resulting after replacement.

variable_mapping(X,L,W)   : To traverse the list X and replace the variables with their corresponding types. The input list is L. The resultant list is W.

replace_address([H,I|T],A,C,[C|Result]) :Custom Replace function which replaces the pair A,B with C in a list where A is either ‘&’ or ‘*’ depending on where this function has been used, B is a type defined by type_define(), and C is “address” or type of variable again depending on usage.

address_of_expr([[A,B]|T],L,O) :This replaces all the pairs &,Var with “address” and outputs it to O.

variable_mapping_address([[A,B,C]|T],L,W): This maps a variable to an address type as well as replaces the pair (*,Var) with its type in the input list.