In the name of God

PL homework #1

Seyed Mohammad Mehdi Ahmadpanah - 9031806

2) Construct a trace of the execution of the following program (i.e. complete the following proof).

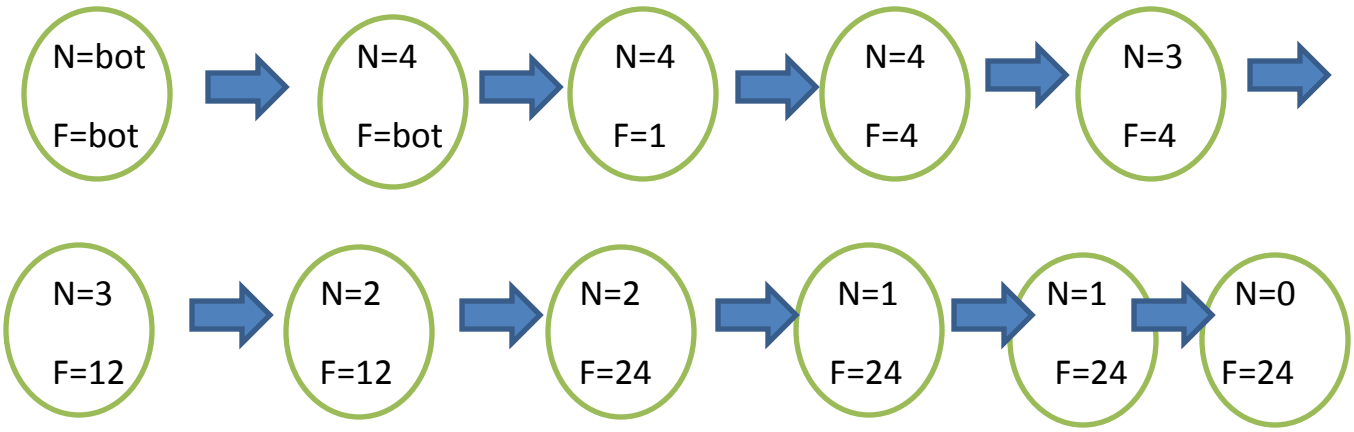
1a . parentOf(john, mary).	Fact
1b . parentOf(kay, john).	Fact
1c . parentOf(bill, kay).	Fact
2. ancestorOf(X,Y) if parentOf(X,Y).	Rule
3. ancestorOf(john,mary).	2,1a,Unification,MP
4. ancestorOf(X,Z) if parentOf(X,Y) and ancestorOf(Y,Z).	Rule
5. ancestorOf(kay,mary).	3,4,1b,Unification,MP
6. ancestorOf(bill,mary).	4,5,1c,Unification,MP
7. not ancestorOf(bill,mary).	Assumption
8. Contradiction.	6,7

3) Construct a trace of the execution of fac(4) given the function definition

1) fac(N) = if N = 0 then 1
   else N*fac(N-1) Definition(Rule)

2) fac(4) = 4 * fac(3) 1,Unification
3) fac(3) = 3 * fac(2) 2,1,Unification
4) fac(2) = 2 * fac(1) 3,1,Unification
5) fac(1) = 1 * fac(0) 4,1,Unification
6) fac (0) = 1 1(Rule)
7) fac(1) = 1 5,6
8) fac(2) = 2 4,7
9) fac(3) = 6 3,8
10) fac(4) = 24 2,9

4) Construct a trace of the execution of the following program
N := 4;
F := 1;
While N > 0 do
F := N*F;
N := N-1;
end;



5) Using the following definition of a list,
list([ ]) -- the empty list
list([X|L]) if list(L) -- first element is X the rest of
the list is L
[X0,...Xn] is an abbreviation for [X0|[...[Xn|[ ]]...]
complete the following computation (proof) and determine the result of
concatenating the two lists.
1. concat([ ],L,L) Fact
2. concat([X|L0],L1,[X|L2]) if concat(L0,L1,L2) Rule
3. ¬concat([0,1],[a,b],L) Assumption

4. ¬concat([1],[a,b],L') 3,2,MT

5. ¬concat([],[a,b],L'') 4,2,MT

6. L''≠[a,b] 5,1,MT

Result = concat([0,1] , [a,b] , [0,1,a,b])

6) Classify the following languages in terms of a computational model: Ada, APL, BASIC, C,COBOL, FORTRAN, Haskell, Icon, LISP, Pascal, Prolog, SNOBOL.

Ada : imperative , APL : functional , BASIC : imperative , C : imperative , COBOL : imperative , FORTRAN : imperative , Haskell : functional(pure) , Icon : imperative , LISP : functional(+imperative) , Pascal : imperative , Prolog : logic , SNOBOL : functional(+logic)

7) For the following applications, determine an appropriate computational model which might serve to provide a solution.
automated teller machine : imperative , flight-control system : imperative , a legal advice service : logic , nuclear power station monitoring system: logic , an industrial robot : functional

9) An extensible language is a language which can be extended after language design time. Compare the extensibility features of C or Pascal with those of LISP or Scheme.
LISP is very extensibility and C also is extensibile.

13) Compare two programming languages from the same computational paradigm with respect to the programming language design principles.
C is strong and good in Implementation , Simplicity , Extensibility , Regularity and Computational Completeness .
JAVA is strong and good in Simplicity , Regularity , Computational Completeness but is not good and weak in Implementation , Extensibility .
C is more efficient and Java is more readable and writable.

14) Construct a program in your favorite language to do one of the following:
a. Perform numerical integration where the function is passed as a parameter.
b. Perform sorting where the less-than function is passed as a parameter.
I choose part “b” :

```
public static void main(String[] arg)

{
```

```
        Vector<Item> item = getItem() ;
        lessthanfunc ltf =  getltf() ;
        Sorting(item , ltf) ;
}
Private void sort(Vector<Record> records , LessThanFunction lessThan)
{
        For(int I = 0 ; I < records.size() ; I ++)
        {
                Record min = records.elementAt(i) ;
                For(int j = I ; j < records.size() ; j++)
                        If(performLessThan(lessThan ,  records.elementAt(j) , min)
                                Min = record.elementAt(j) ;
                Records.swap(min , elementAt(i)) ;
        }
}

private lessthanfunc getltf()
{
        do{
        String str = input.readStreamFully() ;
        }while(SyntaxErr(str)) ;
return new lessthanfunc(str) ;
```

```
}

private String check(lessthanfunc ltf , Item a , Item b)
{
        if(ltf.equals("<"))
                return "a < b" ;
        else if(ltf.equals("="))
                return "a = b" ;
        else if(ltf.equals(">"))
                return "a > b" ;
}
```