```
# ------------------------------------------------

SUBROUTINE AddEdge(row,column)
    GRAPH[X][Y] <- 1
    GRAPH[Y][X] <- 1
ENDSUBROUTINE

# ------------------------------------------------

SUBROUTINE Preparation()

    NODES <- 15
    EDGES <- 14

    # maximum number of friends quest to party
    SizeOfMaxIndSet <- 0

    # number of subsets with size SizeOfMaxIndSet
    # which can provide an alternative set of guests for party

    NumMaxIndSets <- 0

    # array of all node numbers

    FOR I <- 1 TO NODES
        nodes[I] <- I
    NEXT

    # array of friend names corresponding to nodes of Social Graph 2

    names[1] <- "Νίκος"
    names[2] <- "Λυδία"
    names[3] <- "Μαρίλια"
    names[4] <- "Πέτρος"
    names[5] <- "Ελένη"
    names[6] <- "Μάνος"
    names[7] <- "Γιάννης"
    names[8] <- "Άκης"
    names[9] <- "Μαρία"
    names[10] <- "Δημήτρης"
    names[11] <- "Άννα"
    names[12] <- "Αλέξανδρος"
    names[13] <- "Άρτεμις"
    names[14] <- "Ζωή"
    names[15] <- "Αλέκος"
```

```
# Add dislikes edges to graph according to Social Graph 2

    addEdge(1, 2)
    addEdge(1, 8)
    addEdge(3, 6)
    addEdge(4, 5)
    addEdge(5, 6)
    addEdge(5, 9)
    addEdge(5, 10)
    addEdge(6, 8)
    addEdge(7, 8)
    addEdge(9, 12)
    addEdge(10, 15)
    addEdge(11, 12)
    addEdge(11, 15)
    addEdge(13, 14)


ENDSUBROUTINE

# ------------------------------------------------

FUNCTION CheckAllPairsForDislikes(comp_ARRAY, sz , ADJ_MATRIX)

    # Nested loop for all possible pairs of nodes at combination array comp_ARRAY

    FOR I <- 1 TO N
        FOR J <- I TO N
            IF I ≠ J THEN
                IF ADJ_MATRIX[comp_ARRAY[I]][comp_ARRAY[J]] = 1 THEN
                    # dislike edge exists
                    RETURN 1
            ENDIF
        NEXT
    NEXT
    RETURN 0

ENDFUNCTION


# ------------------------------------------------

SUBROUTINE combination1(nodes_ARRAY,NODES,r,index,comp_ARRAY,i)

    IF index = R THEN
        # we have a complete combination with r items
        # from set nodes_ARRAY with NODES times
```

```
        dislikes <- CheckAllPairsForDislikes(comp_ARRAY, r, GRAPH)

        IF dislikes=0 THEN
            IF SizeOfMaxIndSet = THEN
                SizeOfMaxIndSet <- r
                PRINT "Μέγιστος αριθμός καλεσμένων-φίλων στο Πάρτυ : ", SizeOfMaxIndSet
, "\n\n"
                NumMaxIndSets <- NumMaxIndSets + 1
                PRINT "Εναλλακτικό Σύνολο καλεσμένων-φίλων #", NumMaxIndSets , ":\n"

                FOR j <- 1 TO  R
                    PRINT names[comp_ARRAY[j]]);
                NEXT
                PRINT "\n\n"
            ENDIF
        ENDIF

        RETURN
    ENDIF

    # Return when no more elements are there to put in comp_ARRAY[]

    IF I >= NODES THEN
        RETURN
    ENDIF

    # Uses Pascal's identity
    # nCr = (n-1)Cr + (n-1)C(r-1)

    # current is included, put next at next location
    comp_ARRAY[index] <- nodes_ARRAY[i];
    combination1(nodes_ARRAY, NODES, r, index + 1, comp_ARRAY, i + 1);


    #current is excluded, replace it with next
    combination1(nodes_ARRAY, NODES, r, index, comp_ARRAY, i + 1);
    # (Note that i+1 is passed, but index is not changed)


ENDSUBROUTINE

# ------------------------------------------------

SUBROUTINE CalcCombinations(nodes_ARRAY,NODES,r)

    FOR I <- 1 TO r
        combinationArray[I] <- 0
    NEXT
```

```
    combination1(nodes_ARRAY, NODES, r, 0, combinationArray, 0);

ENDSUBROUTINE

# ------------------------------------------------

ALGORITHM FindMaximalIndependentSets()
BEGIN

    # prepare GRAPH adjacency matrix  and other structures
    Preparation()

    # brute force : for all possible subsets of nodes starting with largest subsets
    # we expect subsets of friends having size: 1 < size < 15
    # since we have many dislike edges

    FOR i <- NODES DOWNTO 1
        CalcCombinations(nodes, NODES, r)

        IF SizeOfMaxIndSet ≠ 0 THEN
            RETURN SizeOfMaxIndSet
        ENDIF
    NEXT
    RETURN SizeOfMaxIndSet;

END
```