

```

=====
Node A : - D - C - B
Node C : - A
Node D : - A - B
Node B : - A - D
=====
Apakah Node A dan Edge C terhubung : true
Apakah Node B dan Edge C terhubung : false

Process returned 0 (0x0)   execution time : 0.037 s
Press any key to continue.
|

```

```

Graph.h X Graph.cpp X main.cpp X
1  #ifndef GRAPH_H_INCLUDED
2  #define GRAPH_H_INCLUDED
3  #include <iostream>
4
5  using namespace std;
6
7  #define first(i) i->first
8  #define info(P) P->info
9  #define child(P) P->child
10 #define next(P) P->next
11 #define nextEdge(P) P->nextEdge
12 #define infoEdge(P) P->infoEdge
13
14 typedef char infotype;
15 typedef struct Node *adrNode;
16 typedef struct Edge *adrEdge;
17
18 struct Node{
19     infotype info;
20     adrEdge child;
21     adrNode next;
22     adrNode first;
23 };
24
25 struct Edge{
26     infotype infoEdge;
27     adrEdge nextEdge;
28 };
29
30 adrNode newNode_1301213072(char x);
31 void addNode_1301213072(adrNode &G, adrNode P);
32 adrNode findNode_1301213072(adrNode &G, char x);
33 void addEdge_1301213072(adrNode &G, char x, char y);
34 bool isConnected_1301213072(adrNode G, char x, char y);
35 void printGraph_1301213072(adrNode G);
36
37 #endif // GRAPH_H_INCLUDED
38
39

```

```

Graph.h X Graph.cpp X main.cpp X
1  #include "Graph.h"
2
3  adrNode newNode_1301213072(char x)
4  {
5      adrNode P = new Node;
6      info(P) = x;
7      child(P) = NULL;
8      next(P) = NULL;
9
10     return P;
11 }
12 void addNode_1301213072(adrNode &G, adrNode P)
13 {
14     if (G == NULL)
15     {
16         P = G;
17     }
18     else
19     {
20         adrNode Q = G;
21         while (next(Q) != NULL)
22         {
23             Q = next(Q);
24         }
25         next(Q) = P;
26     }
27 }
28 adrNode findNode_1301213072(adrNode &G, char x)
29 {
30     adrNode Q = G;
31     if (G != NULL)
32     {
33         while (Q != NULL)
34         {
35             if (info(Q) == x)
36             {
37                 return Q;
38             }
39             Q = next(Q);
40         }
41     }
42     else
43     {
44         return NULL;
45     }
46 }
47

```

```

46 }
47 void addEdge_1301213072(adrNode &G, char x, char y)
48 {
49     adrNode Q = findNode_1301213072(G, x);
50
51     adrEdge P = new Edge;
52     infoEdge(P) = y;
53     nextEdge(P) = NULL;
54
55     if(first(G) != NULL)
56     {
57         if(Q != NULL)
58         {
59             if(child(Q) == NULL)
60             {
61                 child(Q) = P;
62             }
63             else{
64                 adrEdge E = child(Q);
65                 while (nextEdge(E) != NULL){
66                     E = nextEdge(E);
67                 }
68                 nextEdge(E) = P;
69             }
70         }
71         else{
72             cout << "Data Parent Tidak Ditemukan" << endl;
73         }
74     } else{
75         cout << "Graph Kosong" << endl;
76     }
77 }
78
79 bool isConnected_1301213072(adrNode G, char x, char y)
80 {
81     adrNode Q = findNode_1301213072(G, x);
82
83     if(Q != NULL){
84         adrEdge P = child(Q);
85
86         while(P != NULL){
87             if(infoEdge(P) == y){
88                 return true;
89             }
90             P = nextEdge(P);
91         }
92     } else{
93         return false;
94     }
95 }
96
97 void printGraph_1301213072(adrNode G)
98 {
99     cout << "===== " << endl;
100     adrNode P = G;
101
102     while (P != NULL){
103         cout << "Node " << info(P) << " : ";
104         if(child(P) != NULL){
105             adrEdge(Q) = child(P);
106             while(Q != NULL){
107                 cout << " - " << infoEdge(Q);
108                 Q = nextEdge(Q);
109             }
110             cout << endl;
111             P = next(P);
112         }
113         cout << "===== " << endl;
114     }
115 }
116

```

Graph.h X Graph.cpp X main.cpp X

```

1 #include "Graph.h"
2
3 int main()
4 {
5     adrNode G;
6     //createGraph_1301213072(G);
7
8     adrNode P = newNode_1301213072('A');
9     addNode_1301213072(G, P);
10    P = newNode_1301213072('C');
11    addNode_1301213072(G, P);
12    P = newNode_1301213072('D');
13    addNode_1301213072(G, P);
14    P = newNode_1301213072('B');
15    addNode_1301213072(G, P);
16
17    addEdge_1301213072(G, 'A', 'D');
18    addEdge_1301213072(G, 'A', 'C');
19    addEdge_1301213072(G, 'A', 'B');
20
21    addEdge_1301213072(G, 'C', 'A');
22
23    addEdge_1301213072(G, 'D', 'A');
24    addEdge_1301213072(G, 'D', 'B');
25
26    addEdge_1301213072(G, 'B', 'A');
27    addEdge_1301213072(G, 'B', 'D');
28
29    printGraph_1301213072(G);
30
31    cout << "Apakah Node A dan Edge C terhubung : ";
32
33    if (isConnected_1301213072(G, 'A', 'C')){
34        cout << "true" << endl;
35    } else{
36        cout << "false" << endl;
37    }
38
39    return 0;
40 }
41

```