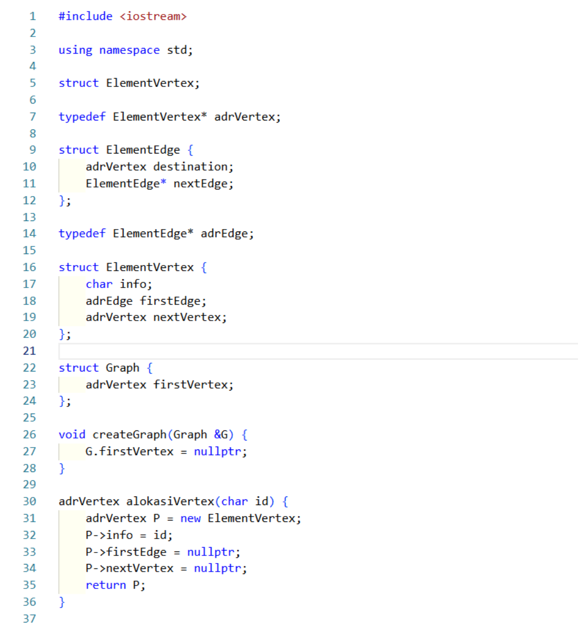
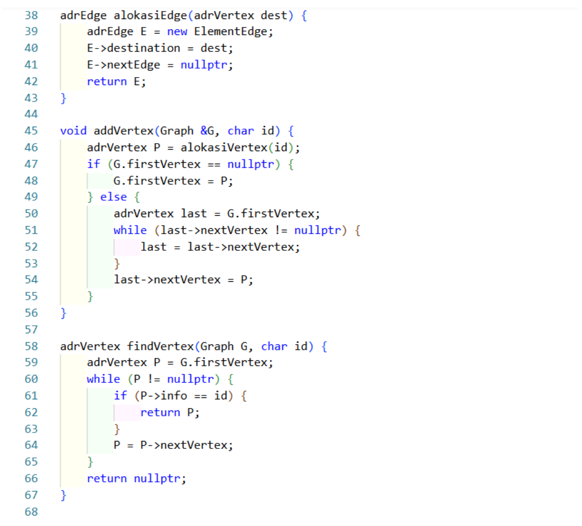
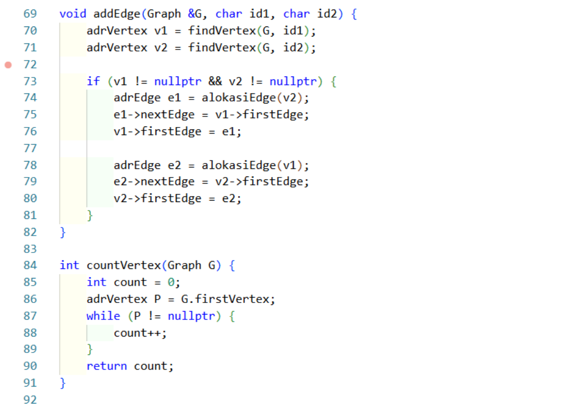
Error Code



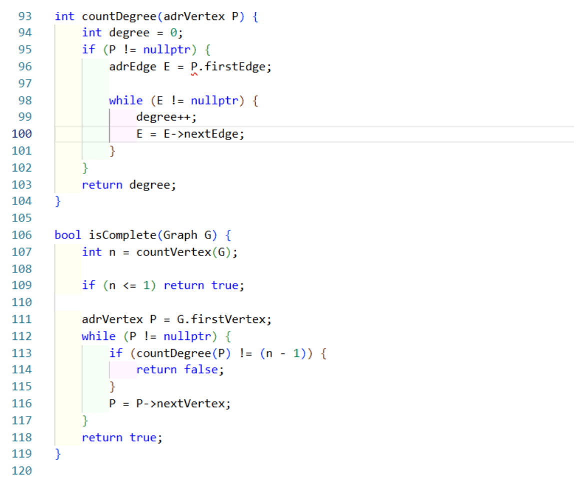
```
1   #include <iostream>
2
3   using namespace std;
4
5   struct ElementVertex;
6
7   typedef ElementVertex* adrVertex;
8
9   struct ElementEdge {
10      adrVertex destination;
11      ElementEdge* nextEdge;
12  };
13
14  typedef ElementEdge* adrEdge;
15
16  struct ElementVertex {
17      char info;
18      adrEdge firstEdge;
19      adrVertex nextVertex;
20  };
21
22  struct Graph {
23      adrVertex firstVertex;
24  };
25
26  void createGraph(Graph &G) {
27      G.firstVertex = nullptr;
28  }
29
30  adrVertex alokasiVertex(char id) {
31      adrVertex P = new ElementVertex;
32      P->info = id;
33      P->firstEdge = nullptr;
34      P->nextVertex = nullptr;
35      return P;
36  }
37
```



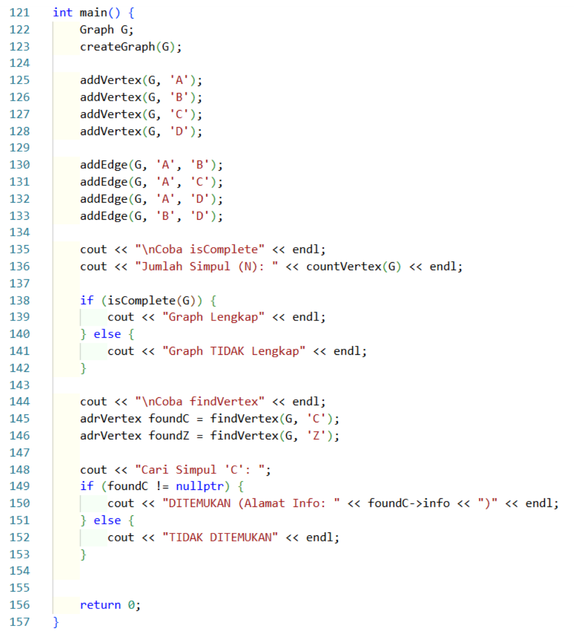
```
38  adrEdge alokasiEdge(adrVertex dest) {
39      adrEdge E = new ElementEdge;
40      E->destination = dest;
41      E->nextEdge = nullptr;
42      return E;
43  }
44
45  void addVertex(Graph &G, char id) {
46      adrVertex P = alokasiVertex(id);
47      if (G.firstVertex == nullptr) {
48          G.firstVertex = P;
49      } else {
50          adrVertex last = G.firstVertex;
51          while (last->nextVertex != nullptr) {
52              last = last->nextVertex;
53          }
54          last->nextVertex = P;
55      }
56  }
57
58  adrVertex findVertex(Graph G, char id) {
59      adrVertex P = G.firstVertex;
60      while (P != nullptr) {
61          if (P->info == id) {
62              return P;
63          }
64          P = P->nextVertex;
65      }
66      return nullptr;
67  }
68
```



```
69  void addEdge(Graph &G, char id1, char id2) {
70      adrVertex v1 = findVertex(G, id1);
71      adrVertex v2 = findVertex(G, id2);
72
73      if (v1 != nullptr && v2 != nullptr) {
74          adrEdge e1 = alokasiEdge(v2);
75          e1->nextEdge = v1->firstEdge;
76          v1->firstEdge = e1;
77
78          adrEdge e2 = alokasiEdge(v1);
79          e2->nextEdge = v2->firstEdge;
80          v2->firstEdge = e2;
81      }
82  }
83
84  int countVertex(Graph G) {
85      int count = 0;
86      adrVertex P = G.firstVertex;
87      while (P != nullptr) {
88          count++;
89      }
90      return count;
91  }
92
```

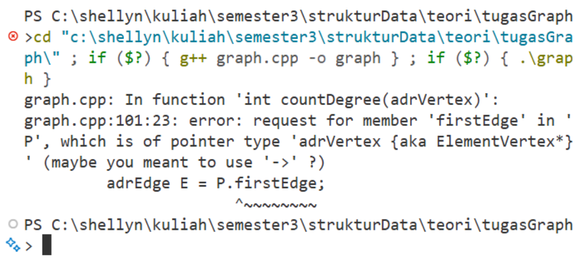


```
93  int countDegree(adrVertex P) {
94      int degree = 0;
95      if (P != nullptr) {
96          adrEdge E = P.firstEdge;
97
98          while (E != nullptr) {
99              degree++;
100             E = E->nextEdge;
101         }
102     }
103     return degree;
104 }
105
106 bool isComplete(Graph G) {
107     int n = countVertex(G);
108
109     if (n <= 1) return true;
110
111     adrVertex P = G.firstVertex;
112     while (P != nullptr) {
113         if (countDegree(P) != (n - 1)) {
114             return false;
115         }
116         P = P->nextVertex;
117     }
118     return true;
119 }
120
```



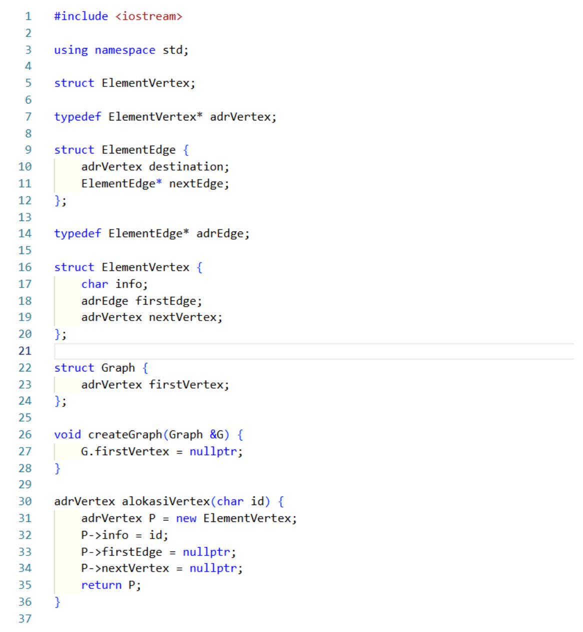
```
121 int main() {
122     Graph G;
123     createGraph(G);
124
125     addVertex(G, 'A');
126     addVertex(G, 'B');
127     addVertex(G, 'C');
128     addVertex(G, 'D');
129
130     addEdge(G, 'A', 'B');
131     addEdge(G, 'A', 'C');
132     addEdge(G, 'A', 'D');
133     addEdge(G, 'B', 'D');
134
135     cout << "\nCoba isComplete" << endl;
136     cout << "Jumlah Simpul (N): " << countVertex(G) << endl;
137
138     if (isComplete(G)) {
139         cout << "Graph Lengkap" << endl;
140     } else {
141         cout << "Graph TIDAK Lengkap" << endl;
142     }
143
144     cout << "\nCoba findVertex" << endl;
145     adrVertex foundC = findVertex(G, 'C');
146     adrVertex foundZ = findVertex(G, 'Z');
147
148     cout << "Cari Simpul 'C': ";
149     if (foundC != nullptr) {
150         cout << "DITEMUKAN (Alamat Info: " << foundC->info << ")" << endl;
151     } else {
152         cout << "TIDAK DITEMUKAN" << endl;
153     }
154
155
156     return 0;
157 }
```

Output:

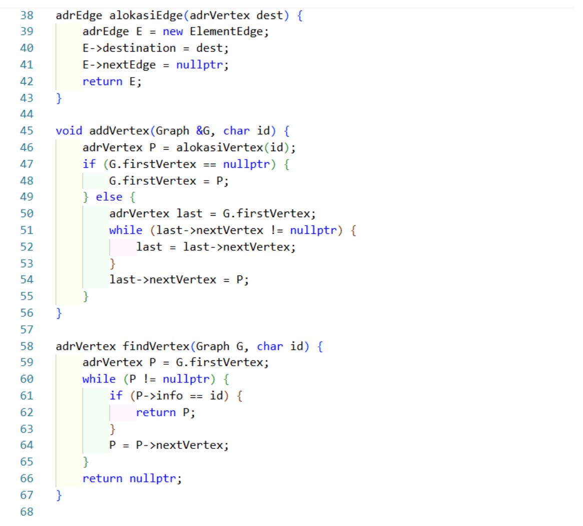


```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasGraph
>cd "c:\shellyn\kuliah\semester3\strukturData\teori\tugasGra
ph\" ; if ($?) { g++ graph.cpp -o graph } ; if ($?) { .\grap
h }
graph.cpp: In function 'int countDegree(adrVertex)':
graph.cpp:101:23: error: request for member 'firstEdge' in '
P', which is of pointer type 'adrVertex {aka ElementVertex*}
' (maybe you meant to use '->' ?)
             adrEdge E = P.firstEdge;
                           ^~~~~~~~~
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasGraph
>
```

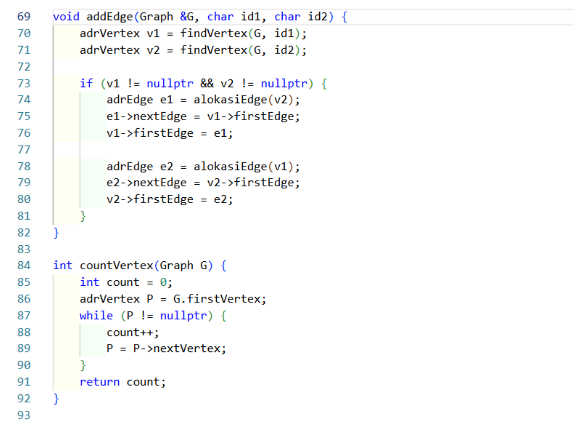
Fix Code



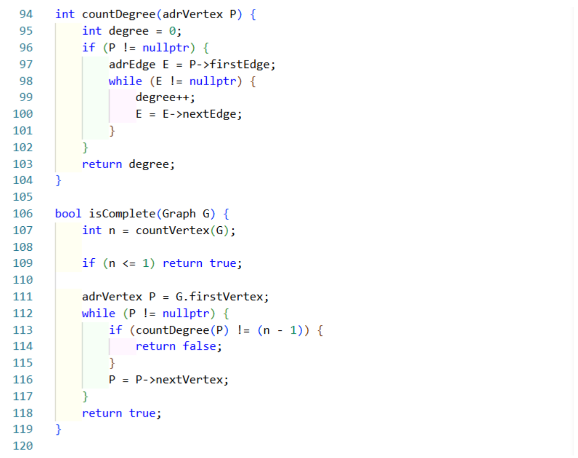
```cpp
#include <iostream>

using namespace std;

struct ElementVertex;

typedef ElementVertex* adrVertex;

struct ElementEdge {
    adrVertex destination;
    ElementEdge* nextEdge;
};

typedef ElementEdge* adrEdge;

struct ElementVertex {
    char info;
    adrEdge firstEdge;
    adrVertex nextVertex;
};

struct Graph {
    adrVertex firstVertex;
};

void createGraph(Graph &G) {
    G.firstVertex = nullptr;
}

adrVertex alokasiVertex(char id) {
    adrVertex P = new ElementVertex;
    P->info = id;
    P->firstEdge = nullptr;
    P->nextVertex = nullptr;
    return P;
}

```



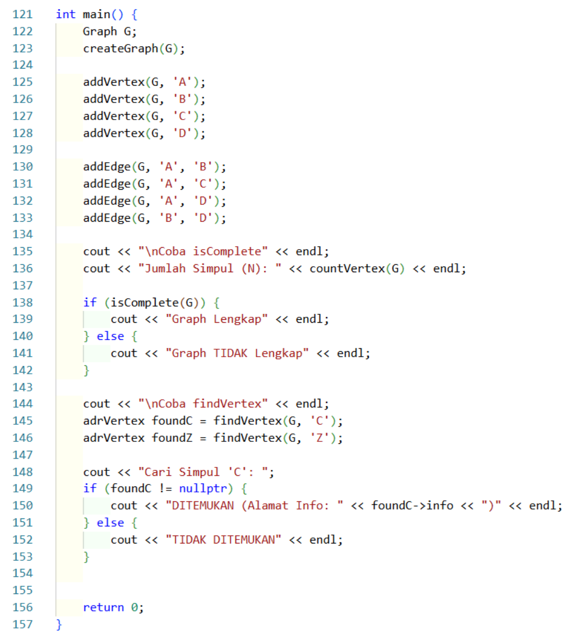
```cpp
adrEdge alokasiEdge(adrVertex dest) {
    adrEdge E = new ElementEdge;
    E->destination = dest;
    E->nextEdge = nullptr;
    return E;
}

void addVertex(Graph &G, char id) {
    adrVertex P = alokasiVertex(id);
    if (G.firstVertex == nullptr) {
        G.firstVertex = P;
    } else {
        adrVertex last = G.firstVertex;
        while (last->nextVertex != nullptr) {
            last = last->nextVertex;
        }
        last->nextVertex = P;
    }
}

adrVertex findVertex(Graph G, char id) {
    adrVertex P = G.firstVertex;
    while (P != nullptr) {
        if (P->info == id) {
            return P;
        }
        P = P->nextVertex;
    }
    return nullptr;
}

```



```cpp
void addEdge(Graph &G, char id1, char id2) {
    adrVertex v1 = findVertex(G, id1);
    adrVertex v2 = findVertex(G, id2);

    if (v1 != nullptr && v2 != nullptr) {
        adrEdge e1 = alokasiEdge(v2);
        e1->nextEdge = v1->firstEdge;
        v1->firstEdge = e1;

        adrEdge e2 = alokasiEdge(v1);
        e2->nextEdge = v2->firstEdge;
        v2->firstEdge = e2;
    }
}

int countVertex(Graph G) {
    int count = 0;
    adrVertex P = G.firstVertex;
    while (P != nullptr) {
        count++;
        P = P->nextVertex;
    }
    return count;
}

```

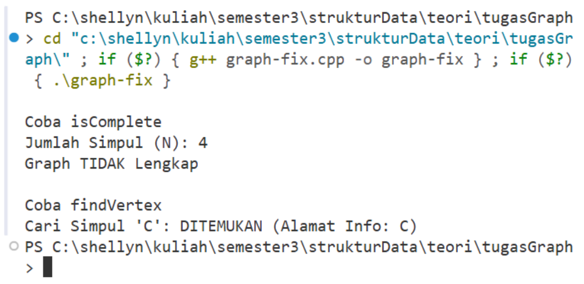


```cpp
int countDegree(adrVertex P) {
    int degree = 0;
    if (P != nullptr) {
        adrEdge E = P->firstEdge;
        while (E != nullptr) {
            degree++;
            E = E->nextEdge;
        }
    }
    return degree;
}

bool isComplete(Graph G) {
    int n = countVertex(G);

    if (n <= 1) return true;

    adrVertex P = G.firstVertex;
    while (P != nullptr) {
        if (countDegree(P) != (n - 1)) {
            return false;
        }
        P = P->nextVertex;
    }
    return true;
}

```



```cpp
int main() {
    Graph G;
    createGraph(G);

    addVertex(G, 'A');
    addVertex(G, 'B');
    addVertex(G, 'C');
    addVertex(G, 'D');

    addEdge(G, 'A', 'B');
    addEdge(G, 'A', 'C');
    addEdge(G, 'A', 'D');
    addEdge(G, 'B', 'D');

    cout << "\nCoba isComplete" << endl;
    cout << "Jumlah Simpul (N): " << countVertex(G) << endl;

    if (isComplete(G)) {
        cout << "Graph Lengkap" << endl;
    } else {
        cout << "Graph TIDAK Lengkap" << endl;
    }

    cout << "\nCoba findVertex" << endl;
    adrVertex foundC = findVertex(G, 'C');
    adrVertex foundZ = findVertex(G, 'Z');

    cout << "Cari Simpul 'C': ";
    if (foundC != nullptr) {
        cout << "DITEMUKAN (Alamat Info: " << foundC->info << ")" << endl;
    } else {
        cout << "TIDAK DITEMUKAN" << endl;
    }


    return 0;
}
```

Output:



```
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasGraph
> cd "c:\shellyn\kuliah\semester3\strukturData\teori\tugasGr
aph\" ; if ($?) { g++ graph-fix.cpp -o graph-fix } ; if ($?)
 { .\graph-fix }

Coba isComplete
Jumlah Simpul (N): 4
Graph TIDAK Lengkap

Coba findVertex
Cari Simpul 'C': DITEMUKAN (Alamat Info: C)
PS C:\shellyn\kuliah\semester3\strukturData\teori\tugasGraph
>
```