```
/*****************************************************************************
**    Program Filename: A9.cpp
**    Author: Jessica Schuler
**    Date: 3-12-14
**    Description: In this game a user needs to navigate through a maze
**    Input: User inputs the direction they would like to go.  They can also
**           get a list of all the locations they have been.
**    Output: Once the user reaches the end of the maze they are informed
**          they finished and a list of their path is output.
*****************************************************************************/

#include<iostream>
#include<vector>
#include<algorithm>
#include<iterator>
#include<limits>

using namespace std;

class node //class node to hold maze locations
{
   public:
            node(); //default constructor
      ~node(); //destructor
      node(char newID); //modifier
      char get_ID(); //function to return ID
      //function to set node directions
      void set_node(node *n, node *s, node *e, node *w);
      //these all return direction name
      node* getNorth();
      node* getSouth();
      node* getEast();
      node* getWest();
   private:
      char ID; //variable for node ID names
      node *North, *South, *East, *West; //variable directions
};

node :: node() : ID(' '), North(NULL), South(NULL), East(NULL), West(NULL)
{
      //left blank
}

node :: node(char newID) :
   ID(newID), North(NULL), South(NULL), East(NULL), West(NULL)
{
      //left blank
}

node :: ~node()
{
      //left blank
}

/*****************************************************************************
**    Function: char get_ID
**    Description: returns ID of the node
**    Parameters:
**    Pre-Conditions:
**    Post-Conditions:
```

```
*****************************************************************************/
char node :: get_ID()
{
      return ID;
}

/*****************************************************************************
**    Function: void set_node
**    Description: assigns direction to each node location
**    Parameters:
**    Pre-Conditions:
**    Post-Conditions:
*****************************************************************************/
void node :: set_node(node *n, node *s, node *e, node *w)
{
      North = n;
      South = s;
      West = w;
      East = e;
}

node* node :: getNorth()
{
      return North;
}

node* node :: getSouth()
{
      return South;
}

node* node :: getEast()
{
      return East;
}

node* node :: getWest()
{
      return West;
}

int main()
{
      //these set up all the nodes of the maze
      node *A = new node('A');
      node *B = new node('B');
      node *C = new node('C');
      node *D = new node('D');
      node *E = new node('E');
      node *F = new node('F');
      node *G = new node('G');
      node *H = new node('H');
      node *I = new node('I');
      node *J = new node('J');
      node *K = new node('K');
      node *L = new node('L');

      //these set up the node directions/locations
      A->set_node(NULL, E, B, NULL);
      B->set_node(NULL, F, NULL, A);
```

```
C->set_node(NULL, G, D, NULL);
D->set_node(NULL, NULL, NULL, C);
E->set_node(A, I, NULL, NULL);
F->set_node(B, NULL, G, NULL);
G->set_node(C, K, H, F);
H->set_node(NULL, L, NULL, G);
I->set_node(E, NULL, J, NULL);
J->set_node(NULL, NULL, NULL, I);
K->set_node(G, NULL, NULL, NULL);
      L->set_node(H, NULL, NULL, NULL);

node* current = A;
std::vector<char> C1; //vector to hold users locations
std::vector<char> C2; //2nd vector to hold user locations
C1.push_back('A'); //sets start point in vector
C2.push_back('A');
int map;

while(current != L) //keeps going until end of maze is reached
{
      cout << "You are in room " << current->get_ID() <<
         " of a maze of twisty passages, all alike! " << endl;
      cout << "You can go: ";

      if(current->getNorth() != NULL) cout << " North. ";
      if(current->getSouth() != NULL) cout << "South. ";
      if(current->getEast() != NULL) cout << "East. ";
      if(current->getWest() != NULL) cout << "West.";

      cout << endl;
      cout << endl;

      cout<<"Do you want a list of rooms you have entered"<<endl;
      cout<<"going back to the start? (Enter 1 for Yes or 2 for No)";
      cin >> map; //gets user input if they want a list
      while(cin.fail()) //checks for numeric entry
      {
         cin.clear();
         cin.ignore(std::numeric_limits<std::streamsize>::max(),'\n');
         cout<<"Invalid Entry! Please enter a Number 1 or 2:";
         cin >> map;
      }

      if(map == 1) //if the user enters 1 for yes
      {
            std::reverse(C2.begin(),C2.end());//reverses vector

            for(int i = 0; i < C2.size(); i++)
            cout<<" "<<C2[i]; //outputs vector
            cout<<endl;
            //reverse vector back so it is reset for next time
            std::reverse(C2.begin(),C2.end());
      }
      else
         cout<<endl;

      cout << "Pick a Direction (N,S,E,W): ";
      char direction;
      cin >> direction; //asks user to pick direction to go
```

```
            if((direction == 'N') && (current->getNorth() != NULL))
               current = current->getNorth();
            if((direction == 'S') && (current->getSouth() != NULL))
               current = current->getSouth();
            if((direction == 'E') && (current->getEast() != NULL))
               current = current->getEast();
            if((direction == 'W') && (current->getWest() != NULL))
               current = current->getWest();

            C1.push_back(current->get_ID()); //adds node ID to vector
            C2.push_back(current->get_ID());
            cout << endl;
      }

      cout << endl;

      cout << "You Finished the Maze!!!" << endl;
      cout << "The Path you took is: ";

      std::ostream_iterator<char>output(cout, " ");
      copy(C1.begin(), C1.end(), output); //outputs user path

      cout << endl;

      //now I delete all new items to release memory
      delete A;
      delete B;
      delete C;
      delete D;
      delete E;
      delete F;
      delete G;
      delete H;
      delete I;
      delete J;
      delete K;
      delete L;

      return 0;
}
```