CSE 100: PA4, B TREES (CONTD)
PA4 (Checkpoint due tonight at 11pm):
Where are you with PA4?

How far along are you with the checkpoint?
A. Haven’t started
B. I have implemented setBoard(), but I don’t know how to go about isOnBoard()
C. I have worked out isOnBoard(), plan to finish by tonight
D. Done
How can you represent the board?
A. vector<Node> board
B. vector<vector<Node>> board
C. Node ** board
D. Node *** board
E. Any of the above and other possibilities
PA4: BogglePlayer

How can you represent the board?
A. vector<Node> board
B. vector<vector<Node>> board
C. Node ** board
D. Node *** board
E. Any of the above and other possibilities

How do you design the class Node?
Depends on what you want to do.

std::vector<int> BogglePlayer::isOnBoard(const std::string &word_to_check)
What do we get from running (vanilla) DFS on this graph, starting with source node $C_5$?

A. All possible words (valid and invalid) starting with $C$
B. A sequence containing all the letters on the board starting with $C$
C. A sequence containing some of the letters on the board starting with $C$
D. Either B or C
What is the simplest change you can make to DFS to get closer to the solution?

DFS(G,v) ( v is the vertex where the search starts )
Stack S := {}; ( start with an empty stack )
for each vertex u, set u.visited := false;
push S, v;
while (S is not empty) do
  u := pop S;
  if (NOT u.visited) then
    set u.visited:=true;
    for each unvisited neighbour w of u
      push S, w;
  end if
end while
END DFS()
isOnBoard(G, v, word)  (v is the vertex where the search starts, word is the word we are looking for)

Stack S := {};  (start with an empty stack), pos := 0
for each vertex u, set u.visited := false;
push S, v;
while (S is not empty) do
  u := pop S;
  if (NOT u.visited AND u.letter == word[pos])
    pos := pos + 1
    u.visited := true;
    for each unvisited neighbour w of u
      push S, w;
  end if
end while
END DFS()

With the modified algorithm, can I find the word CAP (if the source vertex is C5)
A. Yes
B. No
C. Not exactly, but I have made progress
isOnBoard(G,v, word ) ( v is the vertex where the search starts, word is the word we are looking for )

Stack S := {}; ( start with an empty stack ), pos:=0

for each vertex u, set u.visited := false;
push S, v;

while (S is not empty) do
  u := pop S;
  if (NOT u.visited AND u.letter ==word[pos])
    u.visited := true;
    pos:=pos+1
  if pos==length(word)
    return
  for each unvisited neighbour w of u
    push S, w;
  end if
end while
END DFS()

Am I done? What happens if I search for “CANIBAL”?  
A. We can always find it without any problem  
B. We may or may not find it
isOnBoard(G,v, word) (v is the vertex where the search starts, 
word is the word we are looking for)

Stack S := {}; (start with an empty stack), pos:=0
for each vertex u, set u.visited := false;
push S, v;
while (S is not empty) do
u := pop S;
if (NOT u.visited AND u.letter ==word[pos])
  u.visited := true;
  pos:=pos+1
  if pos==length(word)
     return
  for each unvisited neighbour w of u
     push S, w;
end if
end while
END DFS()
Need Backtracking!

<table>
<thead>
<tr>
<th></th>
<th>L 0</th>
<th>A 1</th>
<th>B 2</th>
<th>K 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S 8</td>
<td>A 9</td>
<td>N 10</td>
<td>X 11</td>
<td></td>
</tr>
<tr>
<td>A 12</td>
<td>N 13</td>
<td>X 14</td>
<td>P 15</td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
isOnBoard(G, v, word) (v is the vertex where the search starts, word is the word we are looking for)
Stack S := {}; (start with an empty stack), pos:=0
for each vertex u, set u.visited := false;
push S, v;
while (S is not empty) do
    u := pop S;
    if u.revisit == true
        u.revisit := false
        u.visited := false;
        pos:=pos-1
    else if (NOT u.visited AND u.letter == word[pos])
        u.visited := true;
        pos:=pos+1
        if pos==length(word)
            return
        u.revisit := true;
        push S, u
    for each unvisited neighbour w of u
        push S, w;
end if
end while
END DFS()
```