Linked Stacks and Queues

4

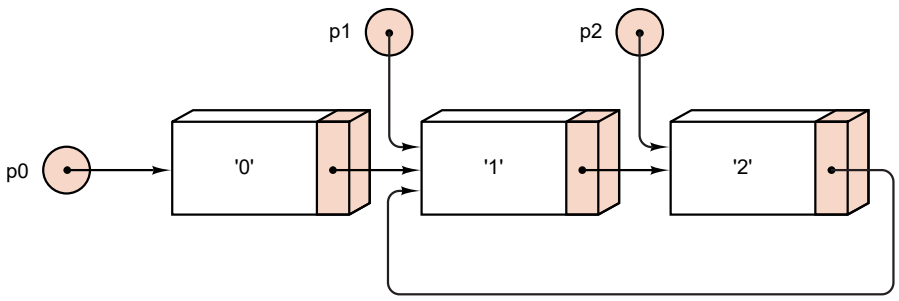
4.1 POINTERS AND LINKED STRUCTURES

Exercises 4.1

E1. *Draw a diagram to illustrate the configuration of linked nodes that is created by the following statements.*

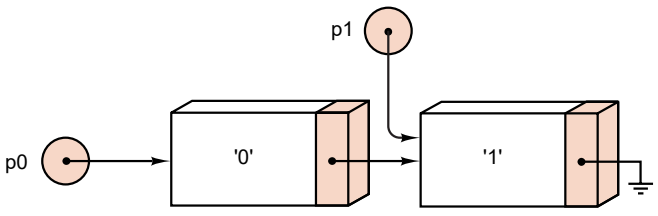
```
Node *p0 = new Node('0');
Node *p1 = p0->next = new Node('1');
Node *p2 = p1->next = new Node('2', p1);
```

Answer



E2. *Write the C++ statements that are needed to create the linked configuration of nodes shown in each of the following diagrams. For each part, embed these statements as part of a program that prints the contents of each node (both* data *and* next*), thereby demonstrating that the nodes have been correctly linked.*

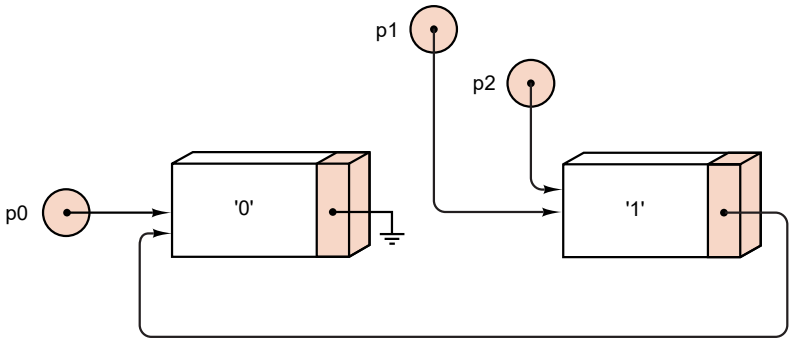
(a)



Answer

```
Node *p0 = new Node('0');
Node *p1 = p0->next = new Node('1');
```

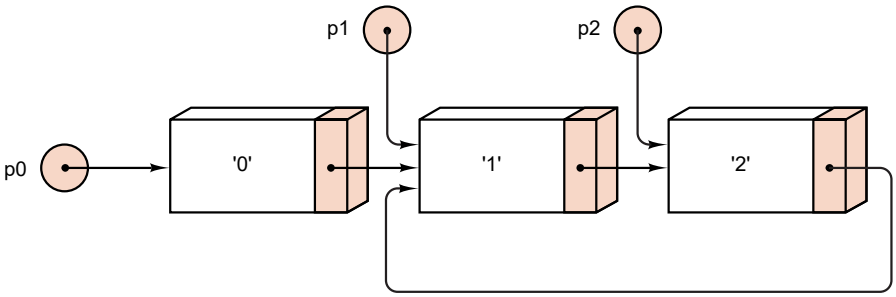
(b)



Answer

```
Node *p0 = new Node('0');
Node *p1 = new Node('1', p0);
Node *p2 = p1;
```

(c)



Answer

```
Node *p0 = new Node('0');
Node *p1 = p0->next = new Node('1');
Node *p2 = p1->next = new Node('2', p1);
```

TOC
Index
Help

4.2 LINKED STACKS

Exercises 4.2

E1. *Explain why we cannot use the following implementation for the method* push *in our linked* Stack.

```
Error_code Stack :: push(Stack_entry item)
{
   Node new_top(item, top_node);
   top_node = new_top;
   return success;
}
```

Answer There is a type incompatibility in the assignment statement top_node = new_top. Moreover, even if we corrected this incompatibility with the assignment top_node = &new_top, the statically acquired node would disappear at the end of the function, and the remaining nodes of the Stack would become garbage.

E2. *Consider a linked stack that includes a method* size. *This method* size *requires a loop that moves through the entire stack to count the entries, since the number of entries in the stack is not kept as a separate member in the stack record.*

```
typedef int Stack_entry;
#include "stack.h"
#include "stack.c"
int ackermann(int m, int n)
/* Pre:  Integers n and k are both at least 0.
   Post: The Ackermann function A(m, n) is returned. */
{
   Stack s;
   s.push(m);
   s.push(n);
   while (1) {
      s.top(n); s.pop();
      if (s.top(m) == underflow) return n; s.pop();
      if (m == 0) s.push(n + 1);
      else if (n == 0) {
         s.push(m – 1);
         s.push(1);
      }
      else {
         s.push(m – 1);
         s.push(m);
         s.push(n – 1);
      }
   }
}
```

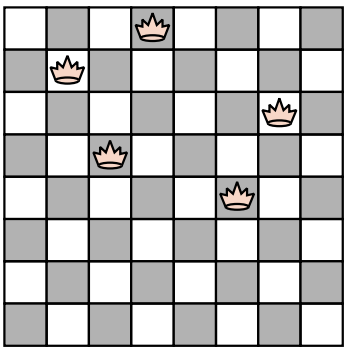
5.3 BACKTRACKING: POSTPONING THE WORK

Exercises 5.3

E1. *What is the maximum depth of recursion in the function* solve_from*?*

Answer The maximum depth of recursion occurs when the solution for the eight-queens problem has been found. Hence the maximum depth of recursion is eight.

E2. *Starting with the following partial configuration of five queens on the board, construct the recursion tree of all situations that the function* solve_from *will consider in trying to add the remaining three queens. Stop drawing the tree at the point where the function will backtrack and remove one of the original five queens.*



Answer