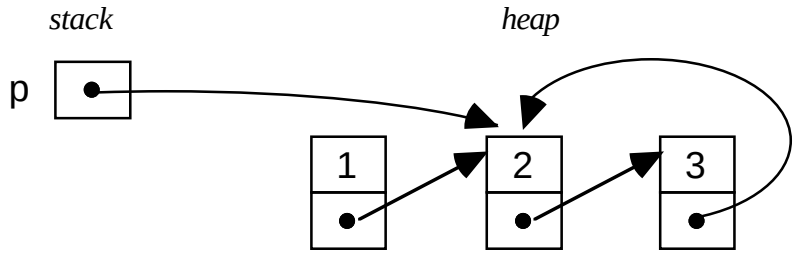
Section Solutions #3

(Note: This handout uses C++ style comments… the //'s after some lines.)

Problem 1: Mini-trace



Problem 2: Recursive ReversePrint

```
static void ReversePrint(cellT *list)
{
   if (list != NULL) {
      ReversePrint(list->link);
      printf("%d ", list->value);
   }
}
```

Note: by changing the order of the recursive call and the `printf` statement, you change whether the list is printed forward or backward.

Problem 3: Append

Note: the first list is passed by reference, since we will change where the pointer points when we hit the final NULL.

```
static void Append(cellT **first, cellT *second)
{
   if (*first == NULL)
      *first = second;
   else
      Append(&(*first)->link, second);
}
```

Problem 4: Stutter

```
void Stutter(cellT *list)
{
   cellT *cur, *newOne;

   for (cur = list; cur != NULL; cur = cur->link) {
      newOne = GetBlock(sizeof(node));     // make new copy of current cell
      newOne->value = cur->value;
      newOne->link = cur->link;     // splice new cell after cur
      cur->link = newOne;
      cur = newOne; // move past new one so don't duplicate again
   }
}
```

Problem 5: RemoveDuplicates (aka Unstutter)

```
void RemoveDuplicates(cellT *list)
{
   cellT *cur, *duplicate;

   for (cur = list; cur != NULL; cur = cur->link) {
      if (cur->link != NULL && cur->value == cur->link->value) { // match!
         duplicate = cur->link;            // record ptr to duplicate one
         cur->link = cur->link->link;      // splice it out
         FreeBlock(duplicate);             // free storage
      }
   }
}
```