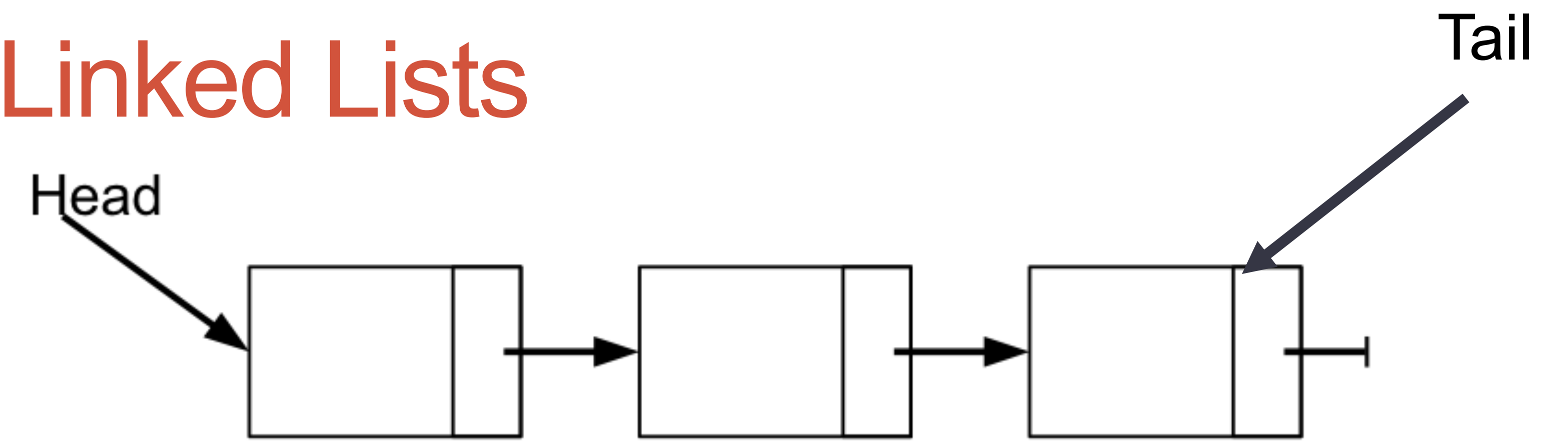
LINKED LISTS

Linked Lists

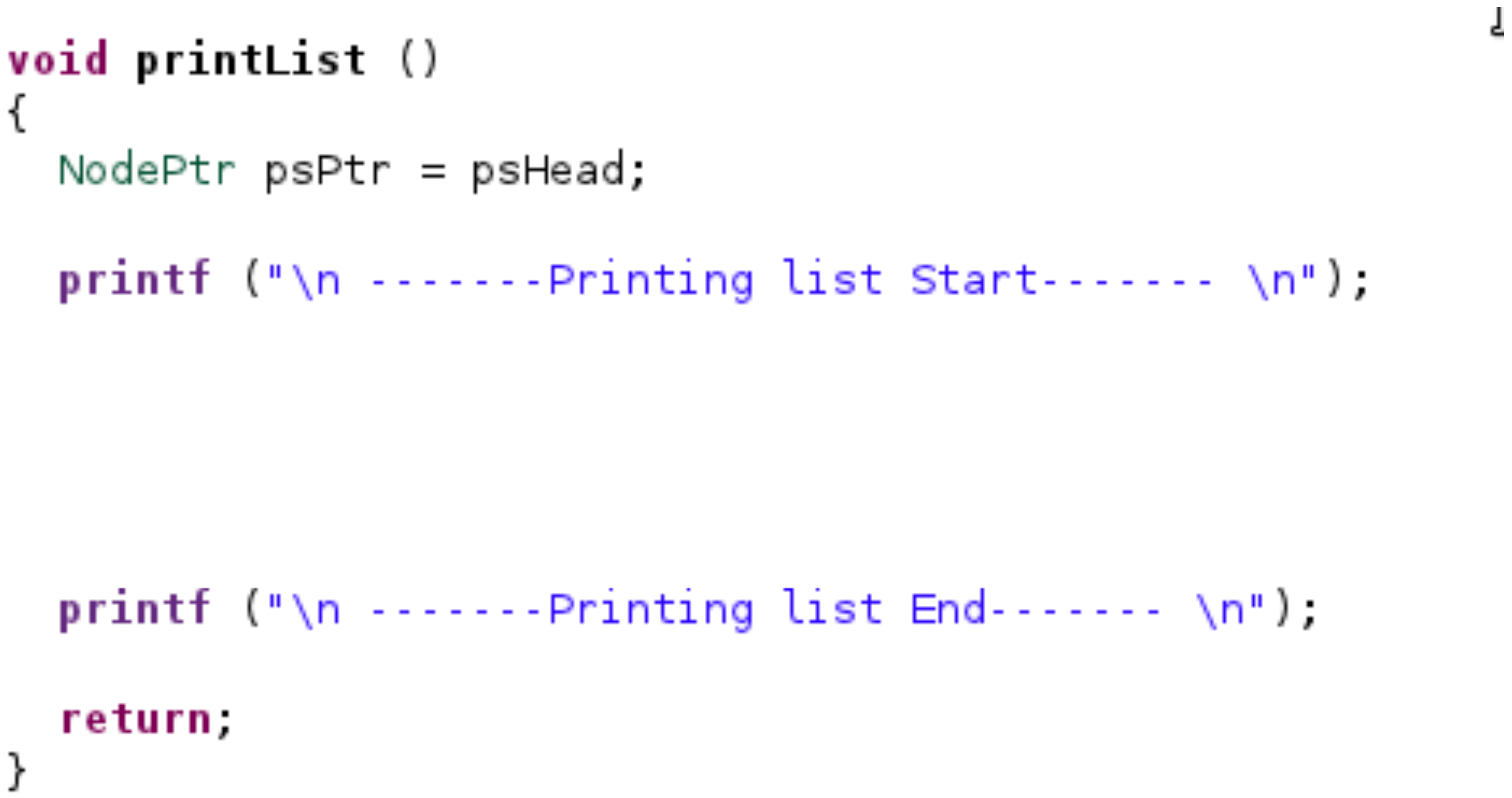


- How would we:
  - Traverse the list and print out every node?
  - Find the length of the list?
  - Insert a node into the list?
  - Search for a particular node and return a pointer to it?
  - Delete a particular node in the list?

Node Representation

```c
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

typedef struct Node *NodePtr;

typedef struct Node
{
  int val;
  NodePtr psNext;
}Node;

NodePtr psHead = NULL;
NodePtr psTail = NULL;
```

Printing the List



```c
void printList ()
{
  NodePtr psPtr = psHead;

  printf ("\n -------Printing list Start------- \n");




  printf ("\n -------Printing list End------- \n");

  return;
}
```

Create List

```
NodePtr createList (int val)
{
  NodePtr psPtr = (NodePtr) malloc (sizeof(struct Node));

  return psPtr;
}
```

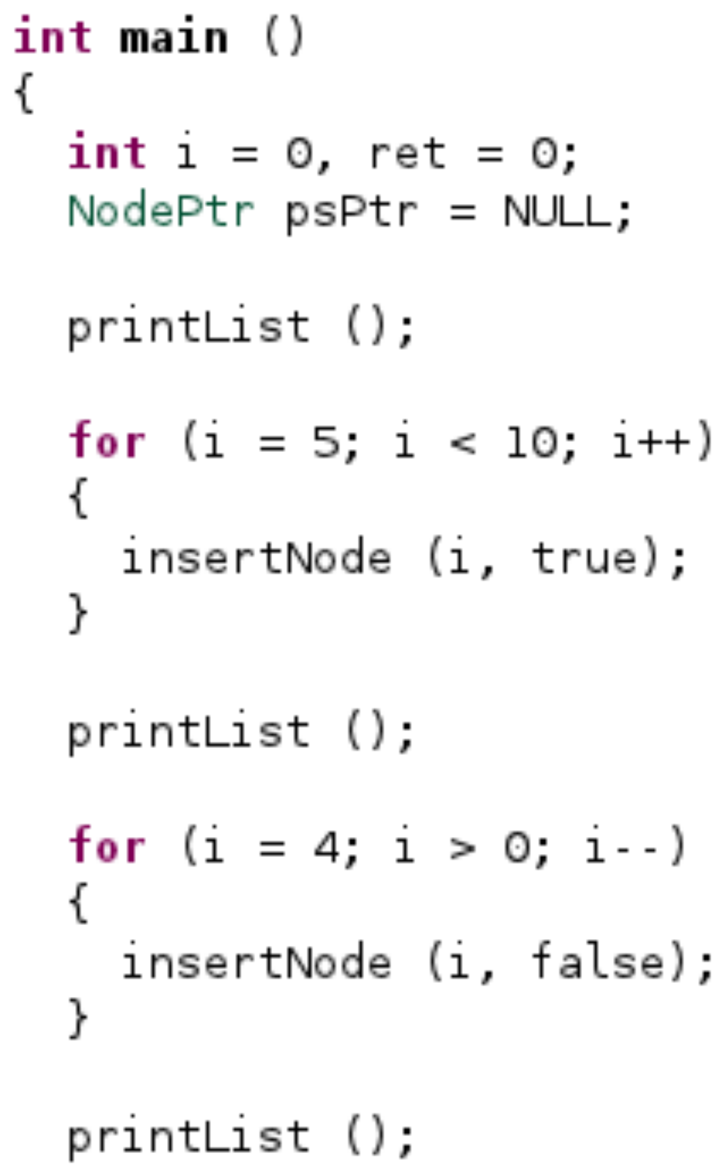
Insert a Node into a List

```
NodePtr insertNode (int val, bool bAddToEnd)
{
  if (NULL == psHead)
  {
    return (createList (val));
  }

  if (bAddToEnd)
  {
    printf ("\n Adding node to end of list with value [%d]\n", val);
  }
  else
  {
    printf ("\n Adding node to beginning of list with value [%d]\n", val);
  }
```

Insert a Node into a List (cont.)

In Main

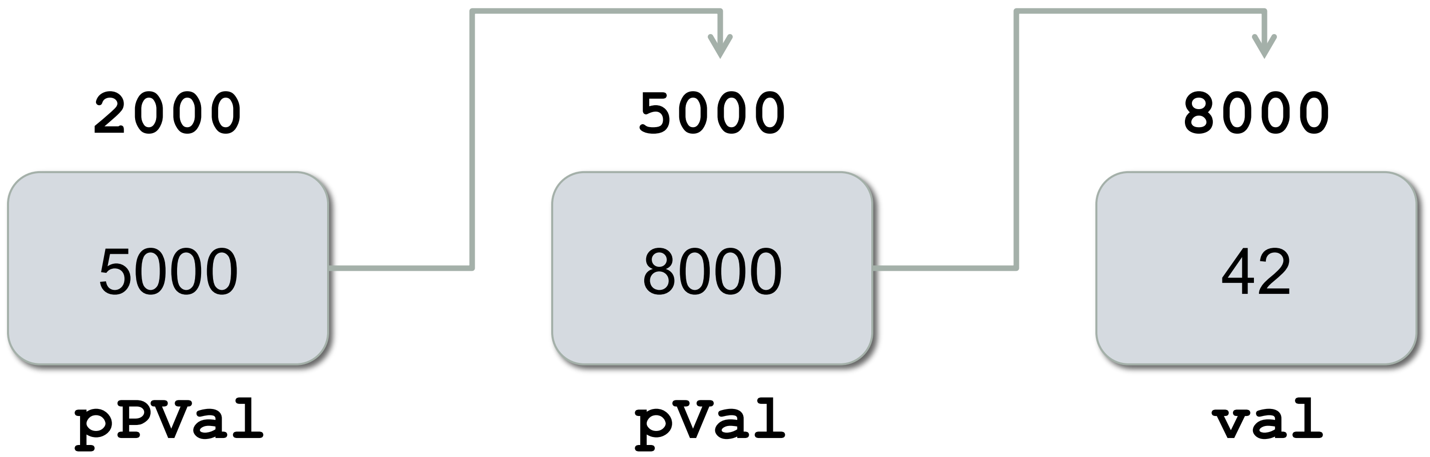
- Create a list and print out all the elements



```
int main ()
{
   int i = 0, ret = 0;
   NodePtr psPtr = NULL;

   printList ();

   for (i = 5; i < 10; i++)
   {
     insertNode (i, true);
   }

   printList ();

   for (i = 4; i > 0; i--)
   {
     insertNode (i, false);
   }

   printList ();
```

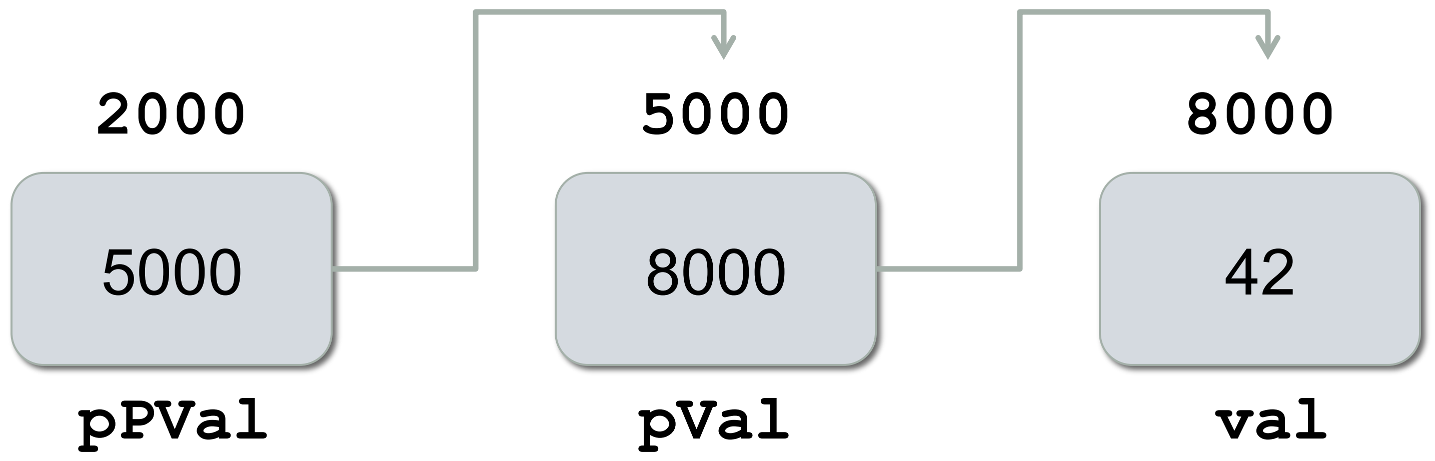
Pointer to a Pointer

- Also referred to as a handle
- Also referred to as a double pointer
- val is an int
- pVal is a pointer to an int
- pPVal is a pointer to a pointer



Pointer to a Pointer

```
int val = 42;
int *pVal = &val;
int **pPVal = &pVal;
```



Pointer to a Pointer

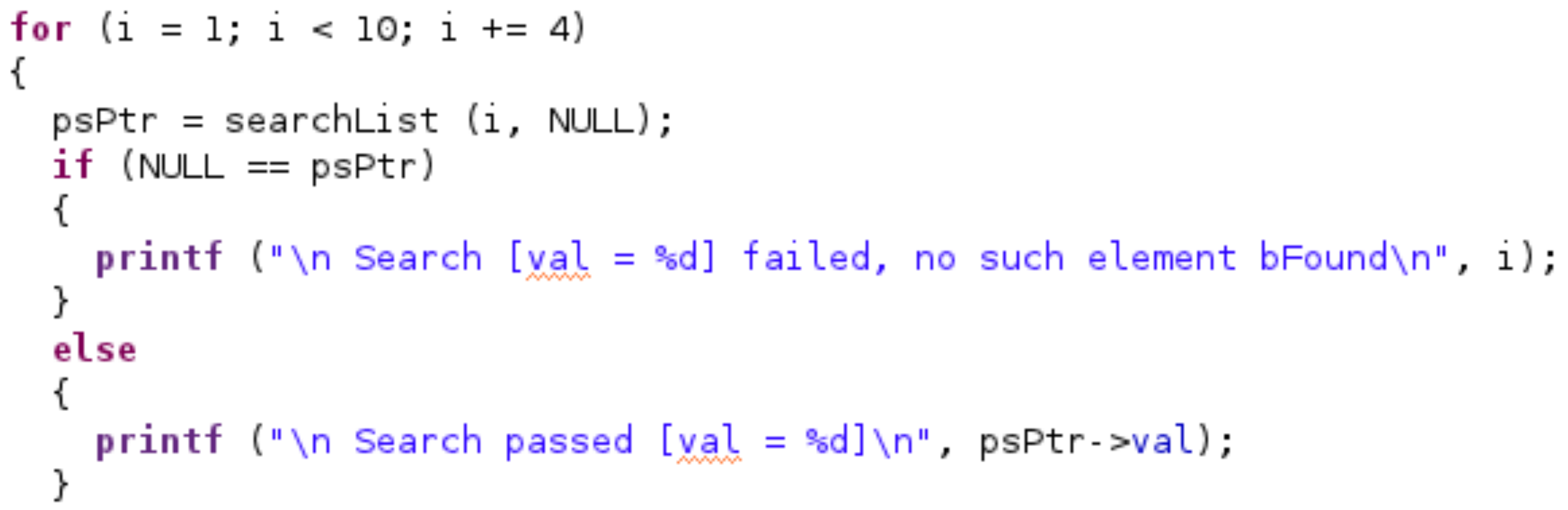
```
int val = 42;
int *pVal = &val;
int **pPVal = &pVal;

val = 99;
*pVal = 22;
**pPVal = 55;
```

Searching a List

```
NodePtr searchList (int val, NodePtr *hPrev)
{
  NodePtr psPtr = psHead;
  NodePtr psTemp = NULL;
  bool bFound = false;

  printf ("\n Searching the list for value [%d] \n", val);
```

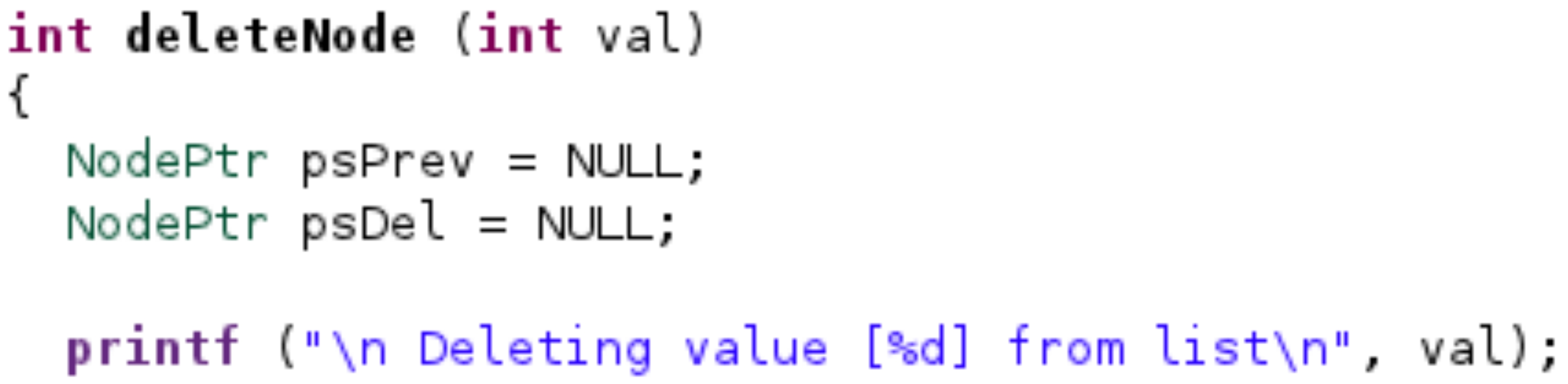
Searching a List (cont.)

In main



```c
for (i = 1; i < 10; i += 4)
{
  psPtr = searchList (i, NULL);
  if (NULL == psPtr)
  {
    printf ("\n Search [val = %d] failed, no such element bFound\n", i);
  }
  else
  {
    printf ("\n Search passed [val = %d]\n", psPtr->val);
  }
```

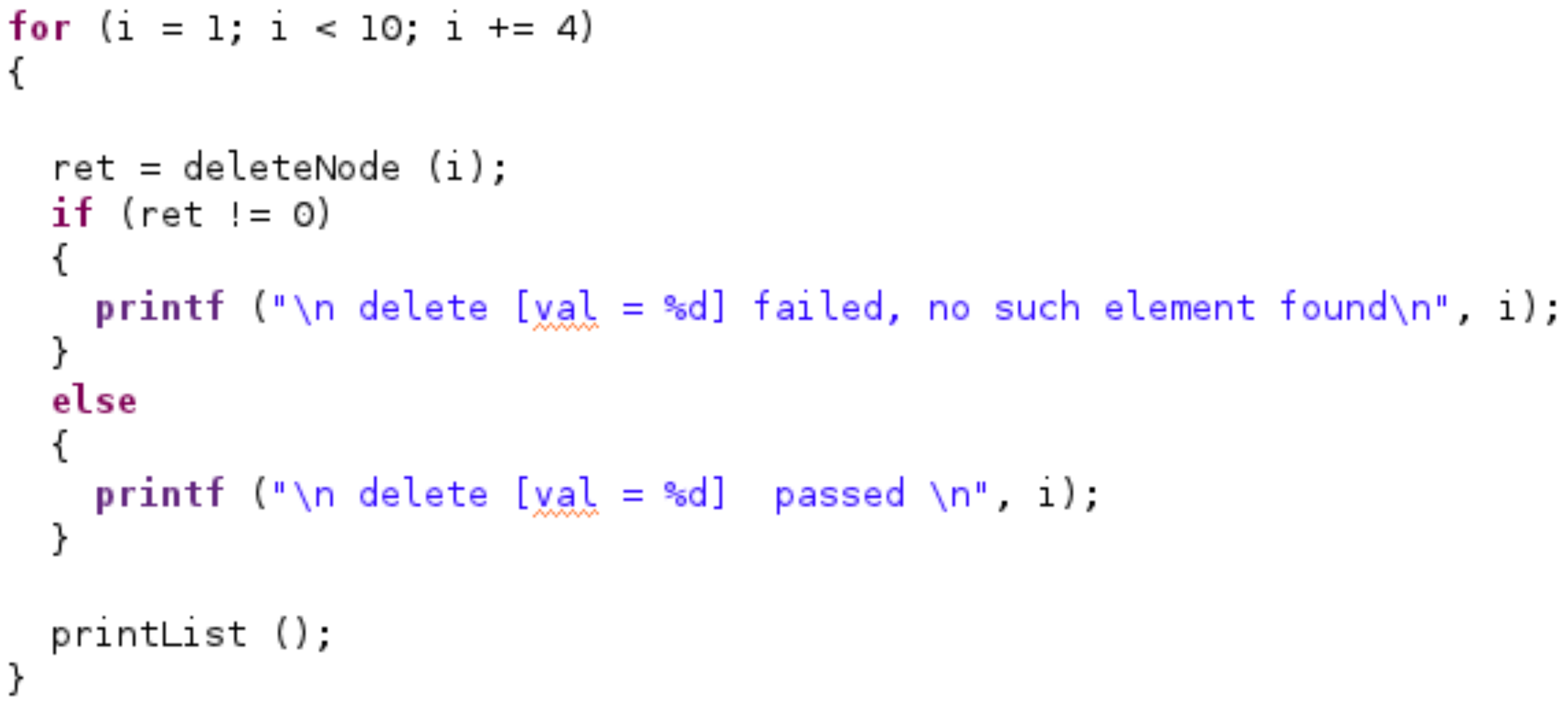
Deleting a Node



```
int deleteNode (int val)
{
  NodePtr psPrev = NULL;
  NodePtr psDel = NULL;

  printf ("\n Deleting value [%d] from list\n", val);
```

Deleting a Node (cont.)

In main



```
for (i = 1; i < 10; i += 4)
{

  ret = deleteNode (i);
  if (ret != 0)
  {
    printf ("\n delete [val = %d] failed, no such element found\n", i);
  }
  else
  {
    printf ("\n delete [val = %d]  passed \n", i);
  }

  printList ();
}
```