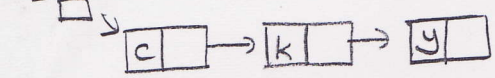
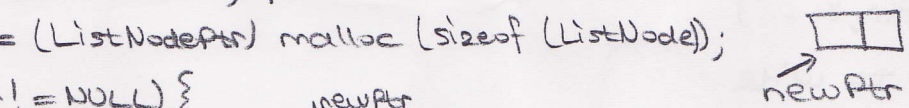
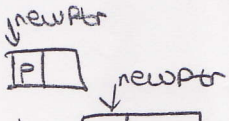
Inserting a Node in Linked List:



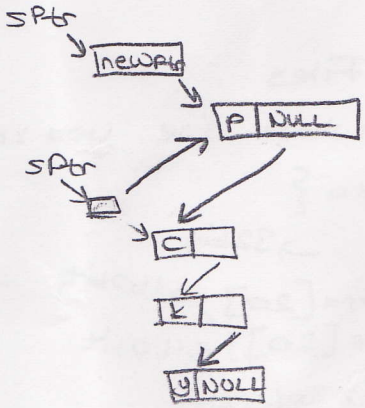
Insert 'p' above.

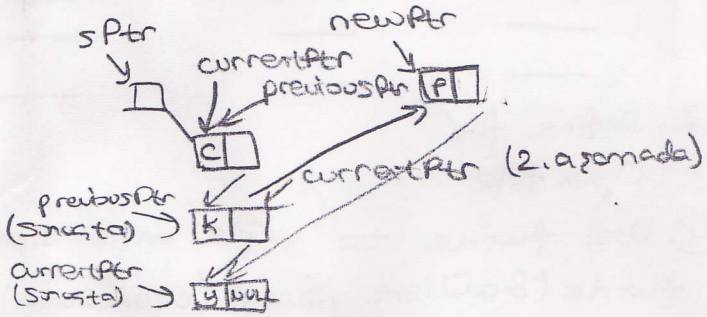
```
void insertNode (ListNodePtr *sPtr, char value)
{ ListNodePtr newPtr, previousPtr=NULL, currentPtr;
  newPtr = (ListNodePtr) malloc (sizeof (ListNode));
if (newPtr != NULL) {
  (*newPtr).data = value;
  (*newPtr).nextPtr = NULL;
  if (*sPtr == NULL) {
     *sPtr = newPtr;
      return;
  }
  if(value <= (**sPtr).data) {
    (*newPtr).nextPtr = *sPtr;
    *sPtr = newPtr;
     return;
  }
  currentPtr = *sPtr;
  while (currentPtr != NULL && value > (*currentPtr).data) {
  previousPtr = currentPtr;
  currentPtr = (*currentPtr).nextPtr;
  }
  (*previousPtr).nextPtr = newPtr;
  (*newPtr).nextPtr = currentPtr;
  }
}
```





sPtr => Start Pointer





```
struct listNode {
   char data;
   struct listNode *nextPtr;
};
typedef struct listNode ListNode;
typedef ListNode *ListNodePtr;
```

Sequential Access Files

1. Define filePtr using
   FILE *filePtr;
2. Open file:
   fopen (filePtr, "w");
3. Use fscanf to read from file:
   fscanf (filePtr, & ...)
4. Use fprintf to write
   fprintf (filePtr, "%...", ...)
5. Close file
   fclose (filePtr);

Random Access Files

1. Use structures whose size you know.

```
struct clientData {
    int accNum;          → 32bit
    char firstName[20];  → 160bit
    char lastName[20];   → 160bit
    double balance;      → 64bit
}
```

Text file:

→ accNum	firstName	lastName	balance
32bit	160bit	160bit	64 ←
———	———	———	——— ←
———	———	———	———

(*) File pointer, 2 clientData ilerlet.

2. Define file
3. Open file
4. Use fwrite to write on a specific record.
   fwrite (&aClient, sizeof (clientData), 1, filePtr);
5. Use fread to read a specific record.
   fread (&aClient, size of (clientData), 1, filePtr);
6. Use fseek to change records:
   fseek (filePtr, 2* size of (clientData), SEEK_SET);
7. Use rewind when necessary.
8. Close the file.