io

read_bytes (infile, uint8_t *buf, nbytes)

wrapper function for read syscall
to keep reading until nbytes.
we read into the buffer

return # bytes read from infile.

write_bytes(outfile, uint8_t *buf, nbytes)

wrapper for syscall write to keep
writing until nbytes
we return how many bytes is written.

read_bit (infile, *bit) * make static buffer
and index

Like read_bytes but now we want to
read a bit each time.
so read a block of bytes each time into buffer
and dole out bits one ata time

write_codes (outfile, Code *c)

buffer each bit into buffer (Loop)
get the bit. If bit is 1 set
If bit is 0 clear

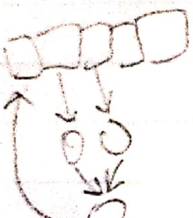
when buffer is Maxed
write_bytes to outfile.

huffman

```
build_tree ( uint64 hist[ static ALPHABET])
        construct a histogram.
        using a priority queue
        create priority queue
              each symbol where frequency > 0
                  create a node and equeue it
        if priority queue size is mor than 1
          -> dequeue 2 nodes
             join them make parent

      return root
```

must be in order
1st = Left
2nd = right

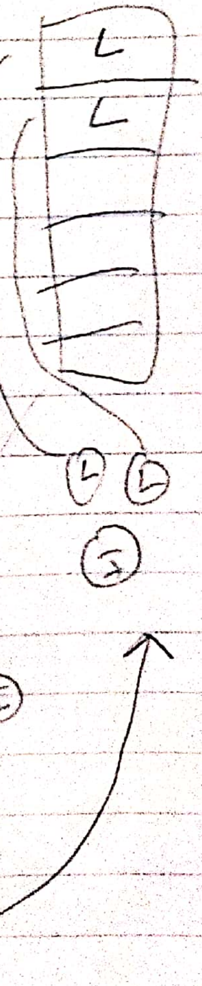


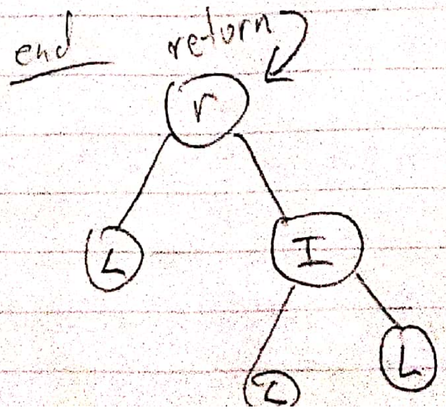
```
build_codes (Noot *root, code table [ALPHABet])
     initalize code
     we check if the root is a Leaf Node
     if it is then asign code to the Node
     else we post order traverse pushing a bit
     building code and popping it.
```

```
dump_tree( outfile, root )

  post orde traversee the huffman tree.
    If at Leaf node
          write 'L' and it's symbol to outfile
    If internal node
          write 'I' to out file


Rebuild_tree( nbytes, tree_dump[nbytes])
  rebuild tree using nbytes as size of array
  Make stack.
     create nodes to push onto stack.
     if Leafs.
  If internal nodes
        pop the Right and then Left
        to make parent.
        push back onto stack.
  return root of tree.
```





end
return
r
L
I
L
L