```c
#include <assert.h>
# include <stdlib.h>
# include <string.h>
# include <stdio.h>
  typedef struct Node Node;
  struct Node{
    char text[32];
    Node *left;
    Node *left;
  };
#define NODE_POOL_CAP 1024
~~Node node_pool [NODE_POOL_CAP];~~
typedef struct{
      size_t sz;
      Node nodes[NODE_POOL_CAP];
   } Node_POOL;

~~static Node_Pool node_pool = {0};~~
static Node_Pool global_node_pool = {0};

Node *node_pool_alloc(Node_Pool *np)
  {
      assert(np->sz < NODE_POOL_CAP);
      Node *result = ~~&~~ &np->nodes[np->sz];
      np->sz += 1;    <- memset(result, 0, sizeof(Node));
      return result;
  }

void node_set_text(Node *node, const char ~~*text~~_cstr)
  {
      size_t n = strlen(text_cstr);
      if (n > sizeof(node->text)-1){
            n = sizeof(node->text)-1;
      }
    memset(node->text, 0, sizeof(node->text));
    memcpy(node->text, text_cstr, n);
  }
```

```sh
#!/bin/sh
set -xe
clang -Wall -Wextra -o main main.c
```

=

```
$ ./build.sh && ./main
+ clang -Wall -Wextra -o main main.c
```

## What is Relative Pointers?

```
# include <stdio.h>

~~int main()~~ typedef struct Node Node;
~~{~~ ~~typedef~~ struct~~{~~ Node{
        char text[32];
        Node * left.
        Node * right;
    };
#define NODE_POOL_CAP 1024
Node node_pool[NODE_POOL_CAP];

int main(){
    return 0;
}
```

What is the size_t data type in C?

size_t is an unsigned integral data type which is defined in various ~~header file such~~

it's a type which is used to represent the size of object in bytes and is therefore used as the return type by the sizeof operator. it is guaranteed to be big enough to contain the size of the ~~gi~~ bigger object the host system can handle.

=

```
#include <stdio.h>
typedef struct Node Node;
struct Node{
    char text [32];
    size_t Left.        // relative Pointers // ~~uin~~ uint16_t
    size_t right;       // relative Pointers // uint16_t
};
#define NODE_POOL_CAP 1024
Node node_pool[NODE_POOL_CAP];
int main()
{
    Node root.
    node_pool[root->left];
    return0;
}
```

~~Word node~~ pool

```
Node . *node_pool_alloc_with_text (Node_Pool *np, const char *text_cstr)
        Node . *result = node_pool_alloc(np)
        node_set_text(result, text_cstr
        return result;
  }

 int main().
 {
  Node . *root = node_pool_alloc_with_text(&global_node_pool, "Hello world");
  return 0;
 }


void  print_tree ( FILE *stream, Node * node)
   {
       UNIMPLEME:
   }
```

To much      Fuck