ABC by Example: Dynamic Memory (Top-Down Approach)

Some Technical Background

```
@ <stdio.hdr>

global a: int = 2;
global b: int;

fn bar(x: int): int
{
    return x * x;
}

fn foo(y: int): int
{
    return bar(y) + 1;
}

fn main()
{
    local c: int = a + b;

    printf("c = %d\n", foo(c));
}
```

```
MCL:tmp lehn$ ./a.out
c = 5
```

```
@ <stdio.hdr>

global a: int = 2;
global b: int;

fn bar(x: int): int
{
    return x * x;
}

fn foo(y: int): int
{
    return bar(y) + 1;
}

fn main()
{
    local c: int = a + b;

    printf("c = %d\n", foo(c));
}
```


```
MCL:tmp lehn$ ./a.out
c = 5
```

```
@ <stdio.hdr>

global a: int = 2;
global b: int;

fn bar(x: int): int
{
    return x * x;
}

fn foo(y: int): int
{
    return bar(y) + 1;
}

fn main()
{
    local c: int = a + b;

    printf("c = %d\n", foo(c));
}
```


```
MCL:tmp lehn$ ./a.out
c = 5
```

```
@ <stdio.hdr>
@ <stdlib.hdr>

fn main()
{
    local x, y: -> int;

    x = malloc(4);
    y = malloc(4);

    *x = 42;
    *y = 13;
    printf("*x = %d\n", *x);
    printf("*y = %d\n", *y);

    free(x);
    free(y);
}
```


```
MCL:tmp lehn$ abc ex2.abc
MCL:tmp lehn$ ./a.out
*x = 42
*y = 13
```

```
@ <stdio.hdr>
@ <stdlib.hdr>

fn main()
{
    local x, y: -> int;

    x = malloc(4);
    free(x);
    y = malloc(4);

    *x = 42;
    *y = 13;
    printf("*x = %d\n", *x);
    printf("*y = %d\n", *y);

    free(y);
}
```


```
MCL:tmp lehn$ abc ex2.abc
MCL:tmp lehn$ ./a.out
*x = 13
*y = 13
```