Exercise
\( (\forall x \forall y \quad y x \rightarrow y z) = \lambda y \quad (\forall x \quad y x \rightarrow y z) \)

Exercise
Exercise
Exercise
```c
{ 
    return
    printf("%d\n",*pa,q);
    *pa = q;
    q = *pa;
    temp = *pa;
    *pa = temp;
)
}

void function2(int pa, int q) {
}
Exercise
Example

Array names and pointers may often be used interchangeably.

- The array name is a constant that always points to the first element of the array and its value cannot be changed.
- The name of an array is the address of the first elements (i.e., a pointer to the first element).
```c
printf("%c", str[i]);

for (i = 0; str[i] != NULL; i++)
    str = str + 1;

int i;
char *str;

char string[] = "This is a string!",
Example:
```

You can also index a pointer using array notation

---

More Pointers and Arrays
Dynamic Memory Allocation

- Dynamic memory may be freed during execution.
- Dynamic memory is finite.
- Dynamic allocation is often used to support data structures such as stacks, queues, linked lists, and binary trees.
- Dynamic allocation is required, offering greater flexibility.
- A program may create as many or as few variables as required.
- Dynamic memory allocated memory is determined at runtime.
realloc function

calloc can be changed using the:
The size of memory requested by malloc or

tfree function

Memory is released using the:
calloc function (cleared memory allocation)

malloc function (memory allocation)

Memory is allocated using the:

dynamic Memory Allocation
include <stdlib.h> to use these functions

- Pointer value to coerce it to the proper pointer type
- A cast operator should be used with the returned 
-
- void pointer
- The pointer returned by these functions is declared to
- will be a NULL value
- If memory can not be allocated, the value returned
- memory
- Both functions return a pointer to the newly allocated

malloc and calloc
Example of malloc

To free the space allocated use free()

x = (double *)malloc(sizeof(double));
/* Allocate memory for 6 doubles. */

int *p;
double *x;
int n = 6, m = 4;
free(x);
To free the space allocated use free()

```c
int *p = malloc(sizeof(int) * 4);
/* Allocate memory for 4 integers. */
p
```

```c
free(p);
```

```c
int *p;
int m = 4;
```
To free the space allocated use free()

```c
q = realloc(p, 2 * m, sizeof(int));
/* Change the size of the memory block pointed by p */

p = (int *) malloc(m * sizeof(int));
/* Allocate memory for 4 integers */
p
int *p;
int m = 4;
```

Realloc (change size of memory)