# CSCI 2132
# Software Development

---

## Lecture 29:

## Linked Lists

Instructor: Vlado Keselj

Faculty of Computer Science

Dalhousie University

# Previous Lecture

- Structures, arrow operator
- Dynamic memory allocation: malloc and free
- Heap (Free Store) started

# Heap (Free Store)

- Memory used for dynamic allocation
  - not to be confused with the heap data structure
- Size stored near allocated block, so `free` knows how much to deallocate
- Free blocks are typically kept in a linked list
- `malloc` may need to get more pages from the system using a system call
- Heap allocation has advantages and disadvantages
- Advantage: Control over object lifetime
- Disadvantage: Efficiency in time and memory space

# Efficient Use of Heap

- Make sure there are no memory leaks
- If we use many small memory objects, e.g., char, it is not a good practice to call malloc for each of them
  - memory fragmentation (external and internal)
  - memory and time overhead

# Additional Allocation Functions

- Two functions: `calloc` and `realloc`
- `calloc` used to allocate memory for an array of objects, and clears it (sets to 0)

```
void *calloc(size_t nmemb, size_t size);
```

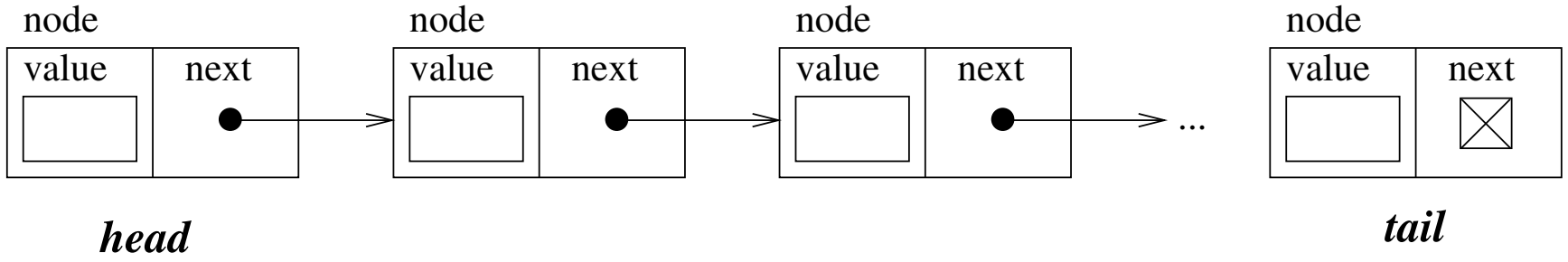
- `realloc` reallocates a block to different size, and preserves relevant content

```
void *realloc(void *ptr, size_t size);
```

- Block location may change, can expand or shrink memory
- `realloc`: given NULL behaves like `malloc`, given size 0 behaves like `free`

# Linked Lists

- A set of nodes linked in one direction
- Advantage over arrays:
  - Insertion and Deletion fast
- Disadvantage:
  - No random access
- Graphical representation:




- In C, use pointers to link elements

# Structure for a Node

- Can be modeled as follows:

```
struct node {
  int value;
  struct node *next;
};
```

- Example, creating empty list:

```
struct node *list = NULL;
```

# Example: list.c

- Fill-in-the-blanks code available at:

  `~prof2132/public/list.c-blanks`