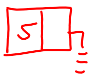
```
typedef struct node Node;
struct node {
      int data;     ] 4 bytes
      Node* next;   ] 4 bytes
};
```

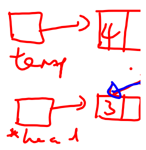
-Create a function
makeNode that returns a node pointer and takes as argument an Integer value and a pointer to a node

```
Node* makeNode (int item, Node* head)
{
  Node* new = (Node*)malloc (sizeof (Node));
  if (new == NULL) {
      return NULL;
  }
  new -> data = item;
  new -> next = head;
  return new,
}
```

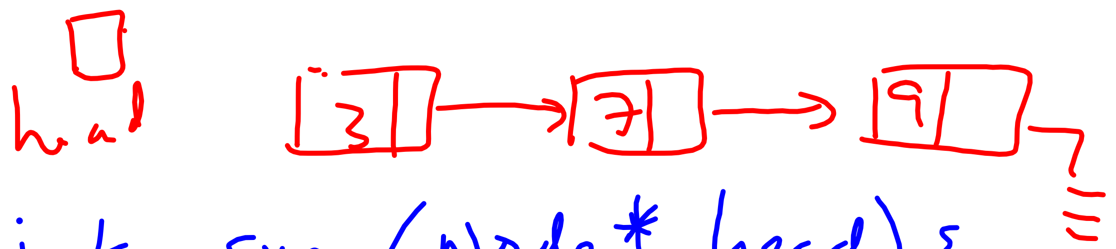
```
Node*  makeNode (int item)
{
  Node* new = (Node*) malloc
                 (sizeof (Node
  If (new != NULL) {
     new -> data = item;
     new -> next = NULL
  }

  return new;
}
```



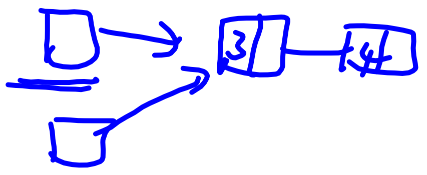
```
Void  insertHead (Node** head, int item
{
  Node* temp = makeNode (item);
  if ( temp == NULL) {
     Printf ("failled to allocate M
     exit (1);
  }
  temp -> next = * head;
  *head = temp;
  return;
}
```

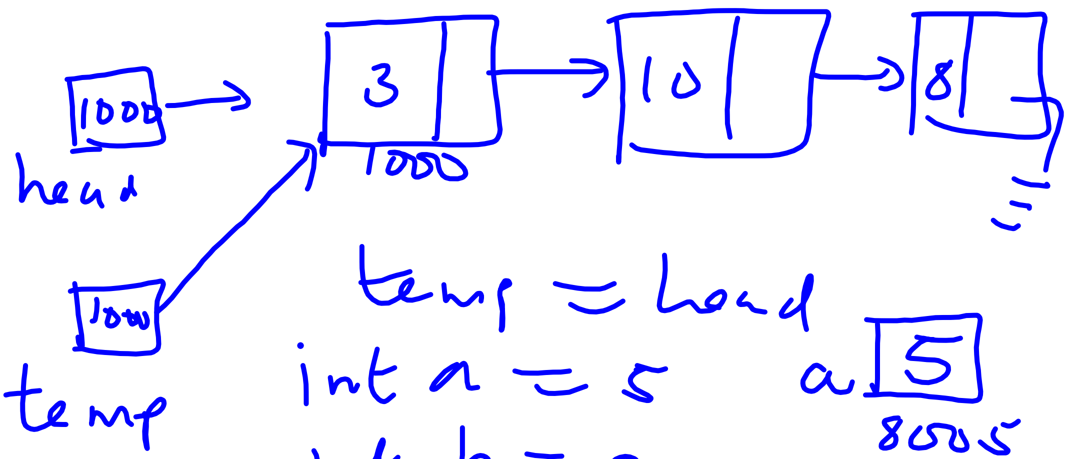


```
int  sum (Node * head)
     * Calculate the sum of all
        data held in the list
```



```
int sum(Node* head){
  int sum = 0;
  Node* temp = head;
  while(temp != NULL) {
    sum = sum + temp->data;
    temp = temp->next;
  }
  return sum;
}
```

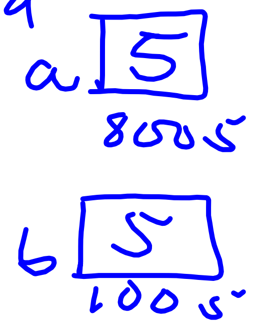




temp = head

int a = 5

int b = a



Recursion

- function calls $\underset{\wedge}{\underline{\text{itself}}}$ simpler instance

→ a loop until a particular condition is met.

- fibonacci
- Tower of Hanoi
→ Exponential function $\underline{\underline{3^4}} = 81$
→ factorial

fact(5) = 120 ✓

fact

$n! = n * n-1 * n-2 * n-n$

$(n-n)! = 1$

// Base case → tells us when to stop

$n! = n * n-1!$

$n-1! = n-1 * n-2!$

```
int factorial(int n) {
   //Base case
    if (n== 0)
              return 1;
   //Recursive case
    return n * factorial(n-1);

}
```

$5! = 5 \times 4 \times 3 \times 2 \times 1 \times 0!$

$5 \times 4 \times 3 \times 2 \times 1 \times 1$

$5! = 5 \times \underline{\overset{24}{4}!}$

$\boxed{120}$. 24 $4! = 4 \times \underline{\overset{6}{3}!}$

$6 = 3! = 3 \times 2!$

$\boxed{2} = 2! = 2 \times 1$

$3^4 = 81$

$= 3 \times 3^{\overset{27}{(4-1)}}$

$3^{4-1} = 3 \times \underline{3^{\overset{9}{(3-1)}}}$

$3^{(3-1)} = 3 \times 3^{\overset{1}{(2-1)}}$

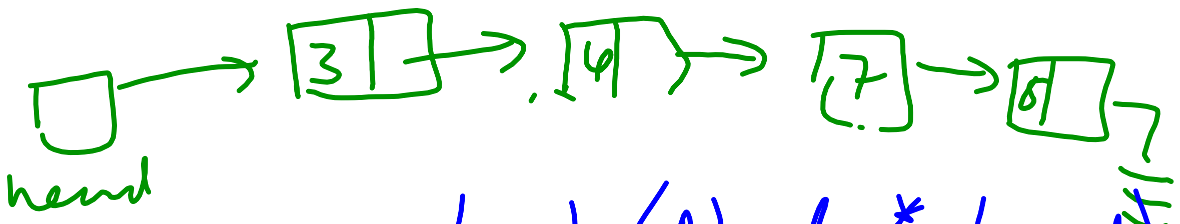
$3^{(2-1)} = 3^{(1-1)}$

1

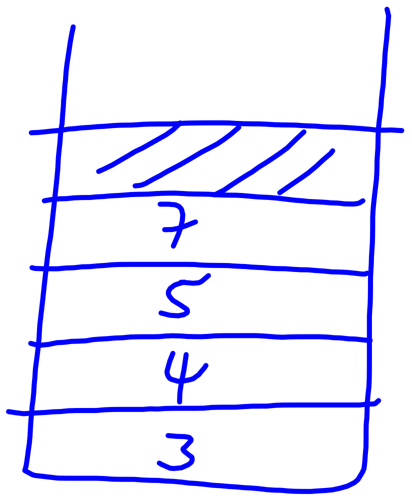
$$f(M^n) = \begin{cases} 1 & \text{if } n = 0 \\ \underline{m * f(m(n-1))} \end{cases}$$

```
int exp(int m, int n)
  if (n==0)
        return 1;
  return m * exp(m,(n-1));
}
```

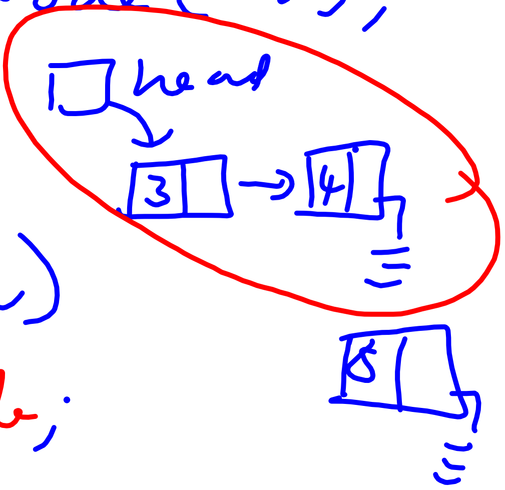
```
int sum(Node * head) {
  // Base case
    if (head == Null)
           return 0;
   // Recursive
return head->data + Sum(head->next);
```



```
void PrintList(Node * head)
  // Base case
    if (!head)
          return;
    Printf("%d \n", head->data);
    PrintList(head->next);
    return;
}
```



```
void tailInsert(Node**head, int n)
  Node* tNode = makeNode(n);
  if(temp == NULL)
        exit(-1);

  if(head == NULL)
        *head = tNode;
  else {
        Node* temp = *head;
        While(temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = tNode;
}
```



```
void destroy(Node** head) {
  if ( *head == NULL) {
         return;
     }
   destroy(& (*head->next));
   free(*head);
}
```

```
void destroy(Node **head) {
→Node *temp = *head;          temp
  while (*head != Null) {
    temp = *head->next;
    free(*head);
    *head = temp;
}
```