ESO207 TA1

Amit Yadav

January 2018

1 n-digit airthmatic ADT

- **Universe**: linked lists of total number of node= n, most significant digit being the last node i.e node: node-¿next=NULL
- ```
  struct node{
      int digit;
      strutc node* next;
  } NUMBER;
  ```
- **Operations**:
  - *constant* 0 (*zero*)
  - *successor* (+1) *function succ*
  - $addition(add): number \times number = number$
  - $multiplication(multiply): number \times number = number$
  - $substraction(subtract): number \times number = number$
  - $division(divide): number \times number = number$
- **Laws**: usual airthmatic laws
- **Exceptions**: Overflows
  - Two n-digit numbers whose addition is more than 99...9(n-times) results results in exception.
  - Two numbers whose multiplication has more than n-digits results in exception.
  - Subtract(smaller, larger) //exception
  - In divide(number1,number2), number2= zero() raises an exception.
  ```

- **Constant 0**

```
NUMBER* zero(){
    for(i=0;i<n;i++){
        temp->digit=0;
        temp=temp->next;
    }
    return temp;
}
```

.

- **Successor**

```
NUMBER* succ(number){
    number=number->digit+1;
    return number;
}
```

- **Addition**

```
NUMBER* add(number1,number2){
    NUMBER* result;
    while(number1 !=NULL && number2 !=NULL){
        result=(number1->digit+number2->digit+carry)\%10;
        carry=(number1->digit+number2_digit+carry)/10;
        number1=number->next;
        number2->next;
    }
    while(number1!=NULL){
        result->next=calloc(1,sizeof(struct node));
        result=result->next;
        result->digit=(number1->digit + carry)%10;
        carry=(number1->digit + carry)/10;
        number1=number1->next;
    }
    while(number2!=NULL){
        result->next=calloc(1,sizeof(struct node));
        result=result->next;
        result->digit=(number2->digit + carry)%10;
        carry=(number2->digit + carry)/10;
        number2=number2->next;
    }
    if(carry==1){
        result->next=calloc(1,sizeof(struct node));
        result=result->next;
        result->digit=1;
    }
```

```
        return head;
    }
```

- **Subtrat(number1,number2)**

```
while(number1!=NULL & number2!=NULL){
    if(number1->digit - carry >= number2->digit){
      result->digit=(number1->digit - number2->digit - carry);
      carry=0;
    }
    else{
      result->digit=(number1->digit - number2->digit - carry + 10);
      carry=1;
    }
    number1=number1->next;
    number2=number2->next;
}
```

Similiarily, for multiply(number1,number2) and divide(number1,number2).

Question 2

2 Queue

- **Universe**: set of linked lists of all sizes size

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

- **Operations**:
  1. **enqueue(queue,val)**: add an integer in the end of the linked list.
  2. **dequeue(queue,val)**: remove the first element from the linked list.
  3. **isEmpty(queue,val)**: returns 1 if the linked list is empty, else returns 0.

Implementation

- **enqueue(queue,val)**:

```
while(queue->next!=NULL){
    queue=queue->next;
}
queue->next=calloc(1,sizeof(struct node));
queue->next->val=val;
printf("%d\n",val);
return temp;
```

- **dequeue(queue,val)**:

```
if(queue==NULL){
    printf("Empty\n");
    return queue;
}
printf("%d\n",queue->val);
queue=queue->next;
```

- **isEmpty(queue,val)**:

```
if(queue==NULL){
    printf("True\n");
    return 1;
}
else{
printf("False\n");
return 0;
}
```

3 Sequence

- **Universe**: Superset of all queues defined above.

```
struct node** sequence;
sequence=calloc(size_of_sequence, sizeof(strict node*));
```

- **Opertaions**

1. **enqueue(val,index)**: Append an element of value val in the end of queue 'index' i.e sequence[index]
2. **dequeue(index)**: Remove the first element from queue 'index' i.e sequence[index]

3. **isEmpty(index)**: Returns 1 if queue 'index' is empty, else return 0.

- **Implementation** Same as queue except that instead of queue, we will be using sequence[index].