C Programming Lab Report

NEETHU S

Roll No: 42
TVE18CS043

May 17, 2019

1 NARCISSISTIC

1.1 Problem Statement

Given a value N, find the difference between number of Narcisstic numbers between 1 and N and between N and N2, both inclusive.
Note : A Narcisstic number or Armstrong number is a number that is the sum of its own digits each raised to the power of the number of digits.

Input Format
A Single Integer N

Constraints

- $1 <= N <= 10^3$

Output Format
An integer giving the difference between the number of Armstrong numbers between 1 and N (inclusive) and between the number of Armstrong numbers between N and N^2 (inclusive).

1.2 Algorithm

Function- **IsAmstrong**

```
1. Start
2. Declare function for finding Armstrong no.
3. Find no. of digits in the input number, num
4. n=num
   (a) Repeat while n not equal to 0.
   (b) Divide n by 10, digit++
5. n=num
6. Repeat while n not equal to 0.
   (a) sum += power of n%10 and digit
   (b) n/=10
7. When sum equals num, return 1
8. Otherwise, return 0
9. Stop
```

Main

```
1. Start
```

```
2. Declare variables N, Ncount, N2count of type long int and i=0,j=N of
   int type.
3. Read N
4. Repeat while i¡=N
    (a) if (IsArmstrong), Ncount++
    (b) i++
5. Repeat while j¡=N*N
    (a) if (IsArmstrong), N2count++
    (b) j++
6. Write Ncount-N2count
7. Stop
```

1.3 Implementaion

The function IsArmstrong checks if the number is Armstrong. Then two loops are run to find the armstrong numbers between 1 and N, and N and N^2. Then the difference of their count(Ncount-N2count) gives the output.

1.4 Program Code

```c
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
int IsArmstrong(long int num){
    long int n, digit=0, sum=0;
    n=num;
    while(n!=0){
        n/=10;
        digit++;
    }
    n=num;
    while(n!=0){
        sum+=pow(n%10,digit);
        n/=10;
    }
    if(sum==num)
        return 1;
    else
        return 0;
}
```

```
int main() {
    long int N,N2;
    int Ncount=0,N2count=0;
    scanf("%ld",&N);
    N2=N*N;
    for(int i=1;i<=N;i++)
        if(IsArmstrong(i))
            Ncount++;

    for(int j=N;j<=N2;j++)
        if(IsArmstrong(j))
            N2count++;

    printf("%d",Ncount-N2count);
    return 0;
}
```

2 SORT THE STRING 1

2.1 Problem Statement

Given a string of english alphabets, display a new string with unique characters that are in the order of frequency in the original string. ie., the character that comes the most number of times should come first in the resultant string. If two characters appear the same number of times, the one that appeared first in the original string should come first in the resultant string.
Input Format
First line gives an integer N, giving the length of the string.
Second line gives the input string.

Constraints

- $1 <= N <= 10^9$

Output Format
The Resultant String.

2.1.1 Algorithm

Function- **Sort**

1. Start
2. Declare function sort.
3. Declare variables i,j,temp[1][2] of int type.
   (a) Starting with the first element(index = 0), compare the current element with the next element of the array.

```
    (b) If the current element is greater than the next element of the array,
        swap them.
    (c) If the current element is less than the next element, move to the
        next element. Repeat Step a.
4. Stop
```

Main

```
1. Start
2. Declare variables i=0,n,j=0,count[26][2] of int type and str[100000000] of
   char type.
3. Initialise array count with 0
4. Read N,str
5. Repeat while str[i]!='\0' and j< 26
    (a) if count[j][0] equals str, count[j][1]++,break
    (b) else if count[j][0] equals 0, count[j][0]=strt[i], count[j][1]++,break
6. sort
7. Repeat while i¡26
    (a) while count[i][0] not equals 0, print count[i][0]
8. Stop
```

2.2 Implementation

A 2 array is used to keep count of each alphabet. Then the array is sorted so that we can arrange the characters by frequency. Then the characters whose count is not zero is printed.

2.3 Program Code

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

void sort(int a[][2], int n) {
     int i, j, temp[1][2];
     for(i = 0; i < n-1; i++) {
         for(j = 0; j < n-i-1; j++)  {
            if(a[j][1] < a[j+1][1]) {
```

```
            temp[0][0] = a[j][0];
            temp[0][1] = a[j][1];
            a[j][0] = a[j+1][0];
            a[j][1] = a[j+1][1];
            a[j+1][0] = temp[0][0];
            a[j+1][1] = temp[0][1];
            }
         }
     }
}
int main() {

    int count[26][2], i, n, j;
    char str[10000000];
    for(i = 0; i < 26; i++) {
        count[i][0] = 0;
        count[i][1] = 0;
    }
    scanf("%d", &n);
    scanf("%s", str);
    for(i = 0; str[i] != '\0'; i++) {
        for(j = 0; j < 26; j++) {
            if(count[j][0] == str[i]) {
                count[j][1]++;
                break;
            } else if(count[j][0] == 0) {
                count[j][0] = str[i];
                count[j][1]++;
                break;
            }
        }

    }
    sort(count, 26);
    for(i = 0; i < 26; i++) {
        if(count[i][0] == 0)
            break;
        printf("%c", count[i][0]);
    }
    return 0;
}
```

3 PALINDROME PAL

3.1 Problem Statement

Given a number N, find the plaindrome number closest to it.
If the number is itself a palindrome, print no pal.
If two adjacent plaindrome numbers are at the same distance from the given number, then print the lower value.

Input Format
A single line having an integer N, which is the input number.

Constraints

- 1 <= N <=10**18

Output Format
A sinlge line having the required output.

3.2 Algorithm

Function- **IsPal**

```
1. Start
2. Declare function for checking Palindrome
3. n=num
    (a) Repeat while n > 0.
    (b) new=new*10 plus n%10
    (c) Divide n by 10
4. When new equals num, return 1
5. Otherwise, return 0
6. Stop
```

Main

```
1. Start
2. Declare variables num,i,j,one,two of type unsigned long long int.
3. Read num
4. i=num+1,j=num-1
5. if IsPal(num), print no pal
```

6. else, check for IsPal for i<10**18 and j>0
    (a) if (IsPal), one=abs(num-i), two=abs(num-j),break
    (b) i++
7. if one>two,print j
8. else if one<two, print i
9. else when one==two, print smallest of i and j
10. Stop

3.2.1 Implementation

3.3 Program Code

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>

int IsPalindrome(unsigned long long int num){

    unsigned long long int n=num, newnum=0;

    while(n>0){
        newnum=newnum*10+(n%10);
        n/=10;
    }
    if(newnum==num)
        return 1;
    else
        return 0;
}

int main() {
    unsigned long long int num,i,j,one,two;
    scanf("%lld",&num);
    if(IsPalindrome(num))
        printf("no pal");
    else{
            for(i=num+1;i<=1000000000000000000;i++)
                if(IsPalindrome(i)){
                    one=abs(num-i);
                    break;
                }
                for(j=num-1;j>0;j--)
                if(IsPalindrome(j)){

                two=abs(num-j);
                break;
            }
        if(one>two)
            printf("%lld",j);

        else if(one<two)
            printf("%lld",i);
        else if(one==two){
            if(i>j)
                printf("%lld",j);
            else
                printf("%lld",i);
            }
        }
return 0;
}
```

4 LIST REVERSE

4.1 Problem Statement

Given a list of numbers, you are to print the numbers in reverse order of input.
Note: Linked List implementation of the problem is required. You are to enter the numbers into a linked list, reverse the linked list and print it.
Input Format
An integer N, giving the number of values Next N lines contain one integer in each line.

Constraints

- $1 <= N <= 10^8$
- $1 <= number <= 100$

Output Format
N lines having one integer in each line.

4.2 Algorithm

Main

1. Start

2. step 2: Define structure num with an integer n,pointer to the same structure *next as members
3. Create 3 nodes, currNode, PrevNode and nextNode.
4. Initialize them as currNode = head; nextNode = null; prevNode = null;
    (a) Repeat while(currNode!=null)
    (b) nextNode = currNode.next;
    (c) currNode.next = prevNode;
    (d) prevNode = currNode;
    (e) currNode = nextNode;
5. At the end set head = prevNode;
6. Stop

4.2.1 Implementation

If there are two or more nodes, then the iterative solution starts with 2 pointers.
reversed: A pointer to already reversed linked list (initialized to head).
list: A pointer to the remaining list (initialized to head->next).
We then set the reversed-¿next to NULL. This becomes the last node in the reversed linked list. reversed will always point to the head of the newly reversed linked list.
At each iteration, the list pointer moves forward (until it reaches NULL).
The current node becomes the head of the new reversed linked list and starts pointing to the previous head of the reversed linked list.
The loop terminates when list becomes NULL and the reversed pointer is pointing to the new head at the termination of the loop.

4.3 Program Code

```
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int num;
    struct node *next;
};
void create(struct node **);
void reverse(struct node **);
void release(struct node **);
void display(struct node *);
```

```
int main()
{
    struct node *p = NULL;
    int n;
    create(&p);
    reverse(&p);
    display(p);
    release(&p);
    return 0;
}

void reverse(struct node **head)
{
    struct node *p, *q, *r;
    p = q = r = *head;
    p = p->next->next;
    q = q->next;
    r->next = NULL;
    q->next = r;
    while (p != NULL)
    {
        r = q;
        q = p;
        p = p->next;
        q->next = r;
    }
    *head = q;
}

void create(struct node **head)
{
    int c, n;
    struct node *temp, *rear;
    scanf("%d", &n);
    while(n)
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        n--;
}
void display(struct node *p)
{
    while (p != NULL)
    {
        printf("%d\n", p->num);
        p = p->next;
    }
}
void release(struct node **head)
{
    struct node *temp = *head;
    *head = (*head)->next;
    while ((*head) != NULL)
    {
        free(temp);
        temp = *head;
        (*head) = (*head)->next;
    }
}
```

5 STRING DECOMPRESSION!

5.1 Problem Statement

Divya and Dheeraj are special agents from the DB agency, Dheeraj intercepted one of the terrorist message which is an alpha numeric code but he couldn't decrypt the information and approaches the brain Divya. Divya is in doubt about the algorithm to implement, Help her to do the algorithm. The algorithm is to convert the alphanumeric code to a code consisting of only alphabets. For example ca2b3 is decrypted as caabbb

Input Format
Compressed string in a single line, msg.

Constraints

- $1 <= length(msg) <= 10^5$

Output Format
Decompressed string

5.2 Algorithm

Main

1. Start
2. Declare variables msg[1000] of char type and i,j,count of int type.
3. Read msg
4. Repeat while msg[i]!='\0'
    (a) if (msg[i] >= 0 and msg[i] <= 9), continue
    (b) else if not msg[i] >= 0 and msg[i] <= 9, print msg[i]
    (c) else count equals count of each character.
5. Stop

5.3 Implementaion

When the character in message is not followed by a digit, then character is printed. Otherwise, the digit is taken as count and the preceeding character is printed count times.

5.4 Program Code

```
#include<stdio.h>
#include<stdlib.h>

int main() {
    int i, j, count;
    char msg[1000];
    scanf("%s", msg);
    for(i = 0; msg[i] != '\0'; i++) {
        if(msg[i] >= '0' && msg[i] <= '9') {
            continue;
        } else if(!(msg[i+1] >= '0' && msg[i+1] <= '9')) {
            printf("%c", msg[i]);
        } else {
            count = atoi(&msg[i+1]);
            for(j = 0; j < count; j++)
                printf("%c", msg[i]);
        }
    }
    return 0;
}
```