SLP_CODING_CHALLENGE_3 – 19/06/2020

QUESTION: 01

Using whatever programming techniques you know, write the cleanest possible function you can think of to print a singly linked list in reverse. The format for the node should be a struct containing an integer value, val, and a next pointer to the following node.

Solution:

```
#include <stdio.h>
struct node
{
	int val;
	struct node* next;
};
void print_reverse( struct node* list )
{
	if ( list != 0 )
	{
		print_reverse( list->next );
		printf( "%d\n", list->val );
	}
}
```

****

QUESTION: 02

Write a program that takes an integer and displays the English name of that value. You should support both positive and negative numbers, in the range supported by a 32-bit integer(approximately -2 billion to 2 billion).

Solution:

```
#include <string>
#include <iostream>
using namespace std;

string num_to_text[] = {
    "", "one", "two", "three", "four", "five",
    "six", "seven", "eight", "nine", "ten",
    "eleven", "twelve", "thirteen", "fourteen",
    "fifteen", "sixteen", "seventeen", "eighteen",
```

```
    "nineteen" };

string tens_to_text[] = { "", "", "twenty", "thirty",
    "forty", "fifty", "sixty", "seventy", "eighty",
    "ninety" };

string power_to_text[] = { "", "thousand", "million", "billion" };

string padIfNeeded (string ans)
{
        if ( ans == "" )
        {
                return "";
        }
        return " " + ans;
}

string translateHundred (int hundred_chunk)
{
        if ( hundred_chunk < 20 )
       {
                return num_to_text[ hundred_chunk ];
        }
        int tens = hundred_chunk / 10;
        int ones = hundred_chunk % 10;
        return tens_to_text[ tens ] + padIfNeeded( num_to_text[ ones ] );
}
string translateThousand (int thousand_chunk)
{
        if ( thousand_chunk < 100 )
        {
                return translateHundred( thousand_chunk );
        }
        else
        {
                int hundreds = thousand_chunk / 100;
                int hundred_chunk = thousand_chunk % 100;
                return num_to_text[ hundreds ]
                    + " hundred"
                    + padIfNeeded( translateHundred( hundred_chunk ) );
        }
}
int main()
{
```

```
    int n;
    cin >> n;
    string number;
    bool is_negative = false;
    if ( n < 0 )
    {
        is_negative = true;
        n *= -1;
    }
    int chunk_count = 0;
    while ( n > 0 )
    {
        if ( n % 1000 != 0 ) {
            number = translateThousand( n % 1000 )
              + padIfNeeded( power_to_text[ chunk_count ]
              + padIfNeeded( number ) );
        }
        n /= 1000;
        chunk_count++;
    }
    if ( number == "" )
    {
        number = "zero";
    }
    if ( is_negative )
    {
        number = "negative " + number;
    }
    cout << number << endl;
}
```

****