Joseph Garcia

3/1/2020

CS 413

Homework 2

1)

a)

// C++ recursive program for checking if a number is palindrome.

#include <stdio.h>

// Function for one digit numbers

int oneDigit(int num)

{

return (num >= 0 && num < 10);

}

// Recursive function for identifying palindromes.

bool Palindrome(int num, int\* dupNum)

{

// Base case

if (oneDigit(num))

return (num == (\*dupNum) % 10);

if (!Palindrome(num/10, dupNum))

return false;

\*dupNum /= 10;

return (num % 10 == (\*dupNum) % 10);

}

// Function uses recursive function Palindrome() to identify palindromes.

int isPalindrome(int num)

{

if (num < 0)

num = -num;

int \*dupNum = new int(num);

return Palindrome(num, dupNum);

}

b)

// Test cases that print results

int main()

{

int n = 0;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

n = 1234554321;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

n = 123454321;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

n = 1221;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

n = 1234;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

n = 7676;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

n = -121;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

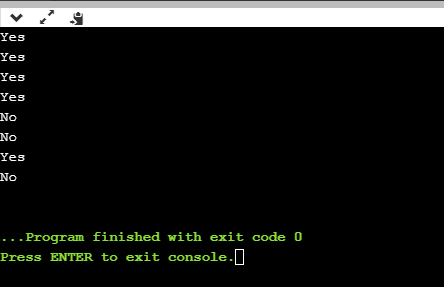
n = -456;

isPalindrome(n)? printf("Yes\n"): printf("No\n");

return 0;

}

1-8)



Test Cases 1-8 from top to bottom

2) Time complexity = O(n) \* O(n) = O(n2)

3) True

T(n) = O(S(n))

There is a N > 0 and M > 0 such that

|T(n)| < MS(n) for n > N

|T(n)| + 2S(n) < MS(n) + 2S(n) = (M + 2)S(n)

|T(n)| + 2S(n) = |T(n) + 2S(n)|

T(n) + 2S(n) = O(S(n))

4)

5)

a) T(n) = T(n-1) + 1

b) T(n) = T(n-1) + 1

T(n-1) = T(n-2) + 1

T(n-2) = T(n-3) + 1

T(n) = T(n-2) + 2

T(n) = T(n-2) + 3

T(n) = T(0) + n = n

T(n) = O(n)

c)

ans = 1;

for i=1 to n do

ans = ans\* i

return ans

Time complexity = O(n)