HOMEWORK 4

NAME: ELSY FERNANDES

STU ID: 1001602253

SECTION 002 (MW 2:30 - 3:50 PM)

TASK 1

Write a Recursion formula and Base case for a function ?

Recursion formula → $T(N) = T(N-1) + C$

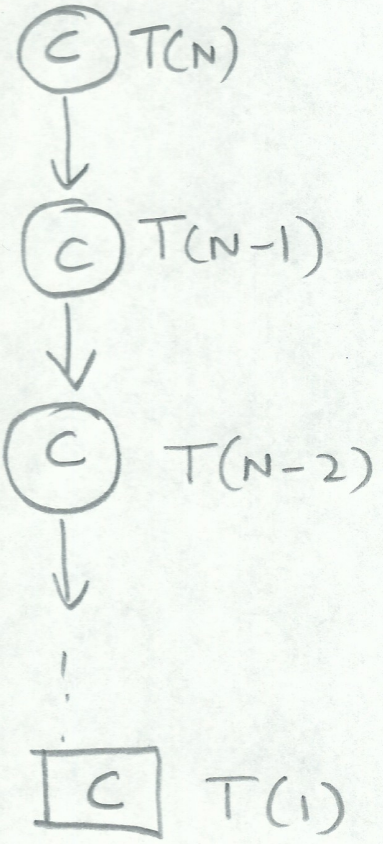
Time complexity of a Recursive call depends on itself in my implementation.

Base case $T(1) = C$

Solving Recursion formula Next Page :→

Solve using Recursion Tree

$T(N) = T(N-1) + C$



level	Avg Pb Size	Cost of Node	Node Per level	level cost
0	N	C	1	C
1	N-1	C	1	C
2	N-2	C	1	C
3	N-3	C	1	C
⋮	⋮	⋮	⋮	⋮
$k = N/1$	1	C	1	C

Tree cost $= C + C + C + C + \cdots C$

$$\frac{NC}{1}$$

$= \underline{\underline{\Theta(N)}}$

NAME: ELSY FERNANDES

Homework 4 – Written answers part

Task 2

Fill in the tables below for each one of the three recursive implementations of smallest.

The remaining empty row is for your convenience in case you run additional tests or want to add comments. Consider the best and worst behavior when these functions are called on an array of 5 elements. The best and worst behavior refers to the number of recursive calls. If there is some array with 5 elements for which one of the functions makes the fewest possible number of recursive calls and another array of the same size for which it makes the largest number of recursive calls, those are you array examples for best and worst case.

If there is no such behavior (it does the same for any data) give one example and say so, but you should still give the counts.

rec_min_1

	Array with 5 elements	base_ct	rec_ct	Max depth	T(N) = ... (for N, not 5)	Solution to T(N) as Θ
Best case	5,4,3,2,1	1	4	5	C+4T(N-1)	N
Worst case	1,2,3,4,5	1	4	5	C+4(T(N-1))	N

rec_min_2

	Array with 5 elements	base_ct	rec_ct	Max depth	T(N) = ... (for N, not 5)	Solution to T(N) as Θ
Best case	5,4,3,2,1	1	4	0	C+4T(N-1)	N
Worst case	1,2,3,4,5	16	15	15	16C+15T(N-1)	N

rec_min_tail

	Array with 5 elements	base_ct	rec_ct	Max depth	T(N) = ... (for N, not 5)	Solution to T(N) as Θ
Best case	5,4,3,2,1	1	5	6	C+5T(N)	N
Worst case	1,2,3,4,5	1	5	6	C+5T(N)	N

Reminder: add written answers for the other tasks as needed.

TASK 3

(4)

```
void PalindromeDecomp(int stringlength, int pos, char * myString, char testArray[100][100], int *
count) {
  int i = pos;
  int z;

  if (pos >= stringlength) {                    //if the index is greater than string length print
the array
    for (z = 0; z < stringlength; z++) {
      if (strlen(testArray[z]) != 0) {
        printf("%s,", testArray[z]);
      }
    }
    ( * count) ++;                              //generate the count of substrings generated
    printf("\n");
    return;
  }
  for (i = pos; i < stringlength; i++) {
    if (checkPalindrome(myString, pos, i)) {    //Function call to check palindrome
      int newSubsLength = i - pos + 1;
      char * newSubstring = malloc(sizeof(char) * newSubsLength);    //if entered string is
palindrome then call a substring function
      newSubstring = substring(myString, pos, i + 1);          //if entered string is palindrome
then call a substring function
      strcpy(testArray[pos], newSubstring);                    //copy the substring generated to test
array
      PalindromeDecomp(stringlength, i + 1, myString, testArray, count);  //palindrome
decomposition recursive call
      strcpy(testArray[pos], "");
      free(newSubstring);
    }
  }
}
```

} → N → Base case

} → C

→ N

→ Recursive Call

} C

Write a Recurrance formula and base case for your function.

$T(N) =$ ____ (Recursive calls) $+$ ____ (local cost)

TASK 3

(4)

Write a Recurrance formula and Base case for your function.

In My code it generate a recursive call Based on String length

If String length = 1 T(1) then it would be

$1\,T(N)$

If Stringlength is N then

$$T(N) = N\,T(N) + CN^2$$

Base Case → CN

COMPILATION INSRUCTIONS:

1. login to omega server using your student id

2. Copy the below files under hw4 folder

    a)palindromic_decomp.c
    b)all.txt
    c)pal_1.txt

3. Compile using below command:-

    gcc -o pal palindromic_decomp.c

4. Run the code without input re direction(Enter inputs manually)

        ./pal

5. Run the code with input re direction for all the three given task

        ./pal<all.txt

6. Run the code with input redirection for task 3

    ./pal<pal_1.txt

7. To Check valgrind memory leaks(Enter inputs manually)

    valgrind --leak-check=yes ./pal

8. To check valgrind memory leaks with input redirection
    valgrind --leak-check=yes ./pal<all.txt
    valgrind --leak-check=yes ./pal<pal_1.txt