CSE 2320
Homework 4 part a, 50 points

Name: Tanmay Sardesai .

For all the programming tasks below write your answer in a C file called **hw4_a.c** .

Task 1 – 12 points Consider this recursive function **foo(N):**

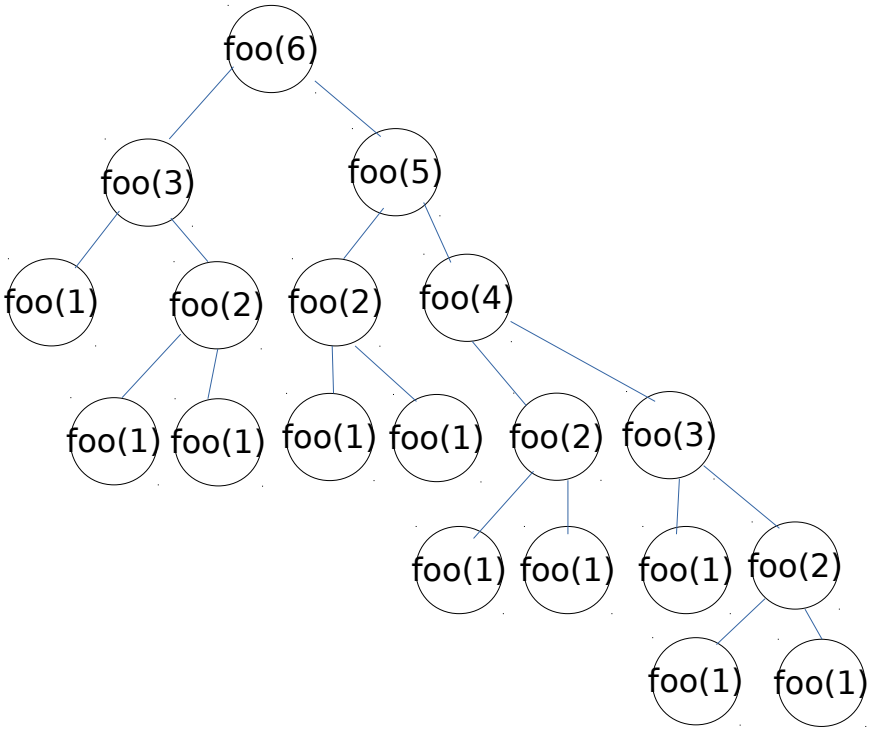
```
int foo(int N){
    if (N <= 1) return 5;
    int res1 = 3*foo(N/2);
    int res2 =   foo(N-1)
    if (res1 >= res 2)
        return res1;
    else
       return res2;
}
```

a) (3 points) Write the recurrence formula for this function (including the base cases) for N>=0. (You do NOT need to solve it.)

foo(0) = 5
foo(1) = 5
foo(N) = max{ 3*foo(N/2) , foo(N-1)}

b) (3 points) Draw the tree that shows the function calls performed in order to compute foo(6) (the root will be foo(6) and it will have a child for each recursive call.)

Answer is on the next page.



c) (6 points) Write this code in hw4_a.c.
(4 points)Re-implement this function with memoization (i.e. use a solution array to look-up and store results of recursive calls).
(2 points) In addition to showing the changes in the foo function, **also write the wrapper function** that calls the memoized **foo** function.

Done in hw4_a.c

Task 2 – 11 points

a) (5 points) Fill-out the edit distance table for the words **`NONSTOP`** and **`ROUND.`**
An empty table is provided below. Fill-out however much you need.

		R	O	U	N	D
	0	1	2	3	4	5
N	1	1	2	3	3	4
O	2	2	1	2	3	4
N	3	3	2	2	2	3
S	4	4	3	3	3	3

T	**5**	**5**	**4**	**4**	**4**	**4**
O	**6**	**6**	**5**	**5**	**5**	**5**
P	**7**	**7**	**6**	**6**	**6**	**6**

Recover the trace for the following edit distance table. The symbols indicate:

\ - diagonal

^ - arrow pointing up

< - arrow pointing left

		r	**e**	**g**	**r**	**e**	**s**	**s**	**i**	**o**	**n**
	<	<	<	<	<	<	<	<	<	<	<
s	^	\	\	\	\	\	\	\	<	<	<
e	^	\	\	<	<	\	<	\	\	\	\
g	^	\	^	\	<	<	<	<	<	<	<
m	^	\	^	^	\	\	\	\	\	\	\
e	^	\	\	^	\	\	<	<	<	<	<
n	^	\	^	^	\	^	\	\	\	\	\
t	^	\	^	^	\	^	\	\	\	\	\

b) (2 points) Show in the table the path you followed (e.g. bold, highlight, circle).

		r	**e**	**g**	**r**	**e**	**s**	**s**	**i**	**o**	**n**
	<	<	<	<	<	<	<	<	<	<	<
s	^	\	\	\	\	\	\	\	<	<	<
e	^	\	\	<	<	\	<	\	\	\	\
g	^	\	^	\	<	<	<	<	<	<	<
m	^	\	^	^	\	\	\	\	\	\	\
e	^	\	\	^	\	\	<	<	<	<	<
n	^	\	^	^	\	^	\	\	\	\	\
t	^	\	^	^	\	^	\	\	\	\	\

c) (3 points) Show all 3 strings: the 2 strings that show the word alignments and the 3rd one showing the cost.

s	**e**	**g**	**m**	**e**	**-**	**-**	**-**	**n**	**t**
r	**e**	**g**	**r**	**e**	**s**	**s**	**i**	**o**	**n**
x	**.**	**.**	**x**	**.**	**x**	**x**	**x**	**x**	**x**

d) (1 point) Using the 3rd string, what is the edit distance between these 2 strings (or the cost of the string alignment)?

The edit distance is number of x's = 7

Task 3 – 8 points

Given an **unlimited number** of the items below:

Item:	A	B	C	D
Weight:	4	6	10	12
Value:	10	21	33	36

a) (3 points) What items will a Greedy algorithm choose for a knapsack of size 16? Show your calculations and justify your answer.

Value/Weight Ratios:

A = 2.5 , B = 3.5, C =3.3, D = 3

So most valuable is B. The most I can fit of B is 2 items i.e 12 weight.

Now I have 16-12 = 4 weight available. The next valuable item that I can fir is A of value 10 and weight 4.

Therefore, with Greedy algorithm :

pick: 2 B and 1 A

weight = 18

value = 52.

b) (5 points) Fill out the solution array, sol, using bottom-up dynamic programming. Follow the style we did in class: for each work-out box (in rows for items A,B,C,D) show the remaining weight and the optimal value obtained by using that item. (You do not have to show the actual calculation to get the

remaining weight. E.g. in the table below row A, problem size 5, shows the remaining weight, 1, and optimal value using item A, 10.)

	0	1	2	3	4	5	6	7	8	9	10	11	12
Sol:	0	0	0	0	10	10	21	21	21	21	33	33	42
Picked	-1	-1	-1	-1	A	A	B	B	B	B	C	C	B
A	-4 -1	-3 -1	-2 -1	-1 -1	0 10	1 10	2 10	3 10	4 20	5 20	6 31	7 31	8 31
B	-6 -1	-5 -1	-4 -1	-3 -1	-2 -1	-1 -1	0 21	1 21	2 21	3 21	4 31	5 31	6 42
C	-10 -1	-9 -1	-8 -1	-7 -1	-6 -1	-5 -1	-4 -1	-3 -1	-2 -1	-1 -1	0 33	1 33	2 33
D	-12 -1	-11 -1	10 -1	-9 -1	-8 -1	-7 -1	-6 -1	-5 -1	-4 -1	-3 -1	-2 -1	-1 -1	0 36

Therefore to backtrack the problem. We pick B then B. i.e 2 B items. So max value is 42 and weight is 12.

Question 4 – 9 points Huffman

(This topic will be covered in the next one or two lectures)

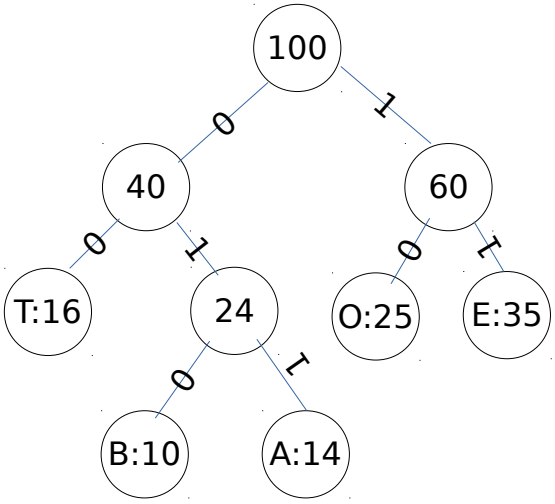
A) (2 points) Can these be valid Huffman codes? If they are, draw the tree, if they are not, justify your answer.

A: 0
B: 01
C: 11
D: 110
E: 111

This is not a Huffman tree because some of the internal nodes don't have 2 children. Example there is no 10 node. Other reason is that the smallest one is not on the left. Example there is no 00 but there is a 01. Also there is no 10 but there is 11.

B) (7 points) What is an optimal Huffman code based on the following set of frequencies: A: 14, B: 10, E:35, O: 25, T: 16 ?

1) (5 points) Build the tree. Show your work at every step. In the tree show the frequencies of each letter and in each node. Type your answer. If not, make sure your drawing is clear.



2) (2 points) Give the Huffman encoding for each letter.

A: 011

B: 010

E: 11

O: 10

T: 00

Question 5 – 9 points Recursion – write code

a) (6 points) Write the code in hw4_a.c
Write a **recursive** function that takes as argument an array and its length and returns 1 if the elements in the array are in increasing order and 0 otherwise. The function must use recursion. You can write auxiliary functions if needed. E.g.:
- for [3,6,1,8] it returns 0.
- for [3, 6, 8, 50, 10000] it returns 1.
- for an empty array it returns 1.

b) (2 points) Write the recursion formula for your function. (Include this answer here or with the code)

rec(0) = 0.

rec(N) = rec(N-1).

recursion ends when either its not finds a non increasing pair or reaches 0.

c) (1 points) What is the runtime of your function? (No justification needed.) (Include this answer here or with the code)

theta(N);