CSE 2320 - Homework 6

NAME:_____

Total points: 115/100 (points past 100 are bonus) Topics: Memoization, Greedy, Dynamic Programming (Knapsack: unbounded, 0/1, fractional)

P1 (4 pts) Given this solution information, for the **unbounded** Knapsack problem below, recover the choices that gave the optimal answer for knapsack capacity 19. Show your work (highlight or circle cells).

```
Item   |   A|   B|   C|   D|
Weight|   3|   4|   7|   8|
Value | the item values are hidden as they should not be used in recovering the solution.
```

8 - 8 = 0 **15 - 7 = 8** **19 - 4 = 15**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
picked				A	B	B	A	C	D	D	A	B	B	A	C	C	D	A	B	B

Items picked for capacity 19: .....

P2 (61 pts) Given the item types below, solve the following problems. Fill in the answer in the table and show your work below.

Item:	A	B	C	D
Weight:	3	4	6	7
Value:	4	7	10	12

	Unbounded Knapsack	0/1 Knapsack	Fractional Knapsack
Dynamic Programming	$$: 24 Items: C, B, B	$$: 23 Items: A, B, D	
Greedy	$$: 21 (DP does better) Items: B, B, B	$$: 23 Items: B, D, A	$$: 24 Items: B, D, half of C

a) (20 pts) Solve the **unbounded** Knapsack problem. Recover the items in the solution and show how you did that (e.g. highlight or circle cells). Show your work as done in class.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Sol	0	0	0	4	7	7	10	12	14	14	17	19	21	22	24
Picked	-1	-1	-1	A	B	B	C	D	B	A	B	B	B	C	B
A, 3, 4	-	-	-	0, 4	1, 4	2, 4	3, 8	4, 11	5, 11	6, 14	7, 16	8, 18	9, 18	10,21	11,23
B, 4, 7	-	-	-	-	0, 7	1, 7	4, 7	3, 11	4, 14	5, 14	6, 17	7, 19	8, 21	9, 21	10,24
C, 6, 10	-	-	-	-	-	-	0, 10	1, 10	2, 10	3, 14	4, 17	5, 17	6, 20	7, 22	8, 24
D, 7, 12	-	-	-	-	-	-	-	0, 12	1, 12	2, 12	3, 16	4, 19	5, 19	6, 22	7, 24

Solution: value = 24, items: C, B, B

b) (20 pts) Solve the **0/1** knapsack problem below (15pts). Use a star to show if the current item was used or not in the solution (8pts). Recover the items in the solution and show how you did that (e.g. highlight or circle cells) (7 pts). Show your work as done in class.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A, 3, 4	0	0	0	4*	4*	4*	4*	4*	4*	4*	4*	4*	4*	4*	4*
B, 4, 7	0	0	0	4	7*	7*	7*	11*	11*	11*	11*	11*	11*	11*	11*
C, 6, 10	0	0	0	4	7	7	10*	11	11	14*	17*	17*	17*	21*	21*
D, 7, 12	0	0	0	4	7	7	10	12*	12*	14	16*	19*	19*	22*	23*

Solution: value = 23, items: A, B, D

c) (8 pts) What items will a Greedy algorithm based on the **ratio**, choose for an **unbounded** knapsack problem of size 14? Show your work.

item	A	B	C	D
ratio	4/3 = 1.34	7/4=1.75	10/6=1.67	12/7=1.71
In order:	B	D	C	A

B: 14-4 = 10

B: 10-4 = 6

C: 6 - 4 = 2 => Solution: value = 3* 7 = 21, items: B,B,B

d) (8 pts) What items will a Greedy algorithm based on the **ratio**, choose for a **0/1** knapsack problem of size 14? Show your work.

B: 14 - 4 = 10

D: 10 - 7 = 3

A: 3 - 3 = 0 => Solution: value = 7+12+4 = 23, items: B,D,A

e) (5 pts) What items will a Greedy algorithm based on the **ratio**, choose for a **fractional** knapsack problem of size 14? Assume you have only one of each item. Show your work.

B: 14 - 4 = 10 => value 7

D: 10 - 7 = 3 = > value 12

C: take half (3) => value: 10*(1/2) = 5 => Solution: value = 7 + 12 + 5 = 24, items: B, D, half of C

P3 (50 pts) Consider this recursive function:

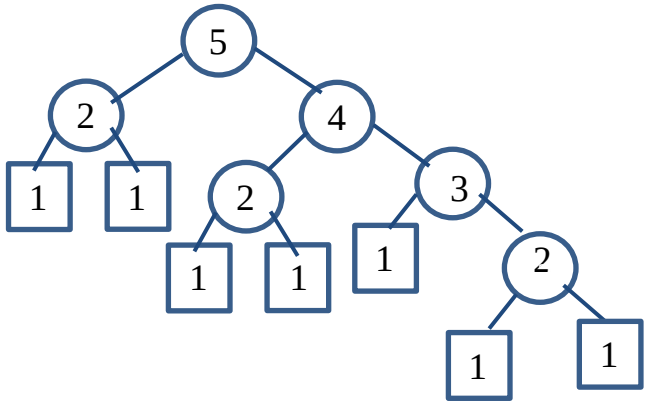
```
int foo(int N){
    if (N <= 1) return 5;
    int res1 = 3*foo(N/2);
    int res2 =   foo(N-1);
    if (res1 >= res 2)
        return res1;
    else
        return res2;
}
```

a) (6 points) Write the recurrence formula for the TIME COMPLEXITY of this function, including the base cases for N>=0. You do NOT need to solve the recurrence. Remember not to confuse the time complexity of the function with what the function calculates.

T(N) = c, for all N ≤ 1

T(N) = T(N/2) + T(N-1) + c

b) (8 points) Draw the tree that shows the function calls performed in order to compute foo(5) (the root will be foo(5) and it will have a child for each recursive call.) Also show what each call returns by using an arrow pointing back from the child to the parent.



In a file called foo.c implement the following functions:
c) (10 pts) `int foo_iterative (int N)` - ITERATIVE solution of this code.
d) (20 pts) … `foo_memoized(…. )` - the MEMOIZED solution. The function signature can be whatever you want. YOU MUST PRINT THE RECURSIVE and BASE CASE CALLS. They should show the N that the function was called for and the text should be indented based on the depth. See Homework 4 for a sample print with indentation based on the depth (min_rec_2_rec_calls_tree.txt) and the code that does that (print_rec_call.txt).
e) (6 pts) `int foo_wrapper(int N)` - a wrapper function that calls the `foo_memoized`.

Your code should not show memory errors with Valgrind. (6 points penalty of it does)
If main does not call your other functions, they cannot be graded and you get 0 points.

Files:

- data1.txt - sample file to be used with input redirection
- sample_run_redirect.txt - sample run with input redirection
- sample_run_user.txt - sample run with user entered data (from keyboard)

Pseudocode for the main function:

print(Enter N:)

```
N <- integer read from user
while (N!=-1){
        res1 = foo_iterative(N)
        print(iterative result: res1)
        res2 = foo_wrapper(N)
        print(memoized result: res2)
        print(Enter N:)
        N <- integer read from user
}
/* NOTE: the code below has a somewhat tricky bug. I challenge you to find it. (It is hard to find
it by just looking at the code.)
However, in an exam you would NOT be penalized for that tricky bug.    */

int foo_iterative(int N){
     int sol[N+1] ;
     int res1, res2, i;
     sol[0] = 5;
     sol[1] = 5;
     for (i = 2; i <= N; i++){
          res1 = 3*sol[i/2];
          res2 = sol[i-1];
          if (res1 >= res2)
               sol[i] = res1;
          else
               sol[i] = res2;
    }
     return sol[N];
}

int foo_wrapper(int N){
    int* sol = malloc( (N+1) * sizeof(int));
    int res, i;
    sol[0] = 5;
    sol[1] = 5;
    for (i = 2; i <= N; i++){
        sol[i] = -1;
```

```
    }
    foo_mem(N, sol, 1);
    res = sol[N];
    free(sol);
    return res;
}

void print_at_depth(int depth, int N){
     int i;
     printf("\n");
     for(i=0;i<depth; i++){
          printf("   ");
     }
     printf("N = %d\n", N);
}

int foo_mem(int N, int* sol, int depth){
     print_at_depth(depth, N);
    if (sol[N] != -1) {
          //printf(" - base case\n");
          return sol[N];
          }
    int res;
    int res1 = 3*foo_mem(N/2, sol, depth+1);
    int res2 = foo_mem(N-1, sol, depth+1);
    if (res1 >= res2)
        res = res1;
    else
        res = res2;
    sol[N] = res;
    return res;
}
```

Remember to include your name at the top.

Write your answers in this document or a new document called **2320_H6.pdf**. It can be hand-written and scanned, but it must be uploaded electronically. Place **2320_H6.pdf** and **foo.c** in a folder called **hw6**, zip that and send it.