COP 3502 11/24/26

Sample Final Sols from Class

2) (10 pts) Write a function that takes in a pointer to a linked list and uses it to create a new linked list that stores the first, third, fifth, etc. values in the input linked list in the original order, returning a pointer to the front of this newly created list. **Write your function recursively.** (For example, if the input list to the function stores 2, 6, 5, 7, 3, 9 then your function will create a new list storing 2, 5, 3, returning a pointer to the node storing 2. If the input list stored 12, 10, 13, then your function will create a new list storing 12, 13, returning a pointer to the node storing 12.) Watch out for NULL pointers!!!

```
typedef struct node {
    int data;
    struct node* next;
} node;

node* oddrankedterms(node* front) {
    if (front == NULL) ~~|| front->next == NULL~~
        return front;
    ~~if (front->next == NULL) {~~
        node* res = malloc(sizeof(node));
        res->data = front->data;
        ~~res->next = NULL;~~
        ~~return res;~~
    ~~}~~ // No more to add
    if (front->next == NULL)
        res->next = NULL;
    else // this is safe.
        res->next = oddrankedterms(res->next->next);

    return res;
}
```

Special case avoids NULL PTR error.

6) (10 pts) Please give **an exact solution** (a function that T(n) satisfies in terms of n) to the following recurrence relation using the iteration technique:

$$T(n) = T(n-1) + n2^n, for\ n > 1$$
$$T(1) = 0$$

$T(n) = T(n-1) + n \cdot 2^n$

$= T(n-2) + (n-1)2^{n-1} + n2^n$

$= T(n-3) + (n-2)2^{n-2} + (n-1)2^{n-1} + n2^n$

$T(n-1) = T(n-2) + (n-1)2^{n-1}$

$T(n-2) = T(n-3) + (n-2)\cdot 2^{n-2}$

After k iterations

$$T(n) = T(n-k) + \sum_{i=n-k+1}^{n} i \cdot 2^i$$

Let $n-k=1$, so $k=n-1$, plug into the recurrence:

$$T(n) = T(1) + \sum_{i=2}^{n} i \cdot 2^i = \sum_{i=2}^{n} i \cdot 2^i$$

Let $S = \sum_{i=2}^{n} i \cdot 2^i$

$S = 2\cdot 2^2 + 3\cdot 2^3 + 4\cdot 2^4 + \dots + n\cdot 2^n$

$-2S = \quad 2\cdot 2^3 + 3\cdot 2^4 + \dots + (n-1)\cdot 2^n + n\cdot 2^{n+1}$

$-S = 2\cdot 2^2 + 1\cdot 2^3 + 1\cdot 2^4 + \dots + 1\cdot 2^n - n\cdot 2^{n+1}$

$-S = (1 + 2^0 + 2^1 + 2^2 + 2^3 + \dots + 2^n) - n\cdot 2^{n+1}$

$-S = 1 + \sum_{i=0}^{n} 2^i - n\cdot 2^{n+1}$

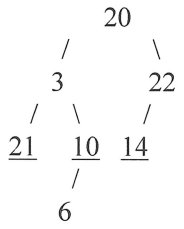
$-S = 1 + 2^{n+1} - 1 - n\cdot 2^{n+1}$

$S = n\cdot 2^{n+1} - 2^{n+1} = (n-1)2^{n+1}$

7) (6 pts) Show the contents of the following array after each merge in Merge Sort.

Original	17	13	7	19	22	2	15	14
After 1st								
After 2nd								
After 3rd								
After 4th								
After 5th								
After 6th								
Last	2	7	13	14	15	17	19	22

8) (10 pts) Consider the following problem: given a pointer to the root of a binary tree, and a non-negative integer, k, find the sum of all the values in the nodes that are exactly k levels below the root node. For example, in the tree below, the sum of the values 2 levels below the root is 21 + 10 + 14 = 45.



Write a function that takes in a pointer to the root of a binary tree and the value of k and returns this sum. Your function **must be recursive.**

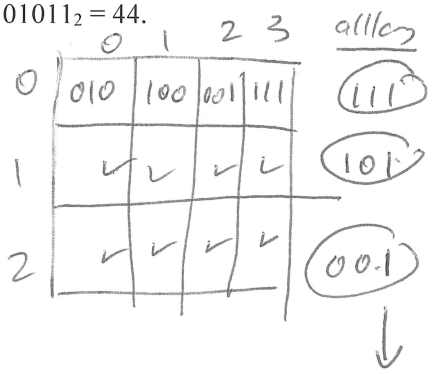
```
typedef struct treenode {
    int data;
    struct treenode* left;
    struct treenode* right;
} treenode;

int sumdepthk(treenode* root, int k) {
    if (root == NULL) return 0;
    if (k == 0) return root->data;
    int res = 0;
    res += sumdepthk(root->left, k-1);
    res += sumdepthk(root->right, k-1);
    return res;
}
```

10) (10 pts) Consider the task of printing out the (in lowercase letters) the **first word** alphabetically stored in a trie. (You may assume if a link exists in a trie, that some word exists further down that link.) Write a **void recursive** function to accomplish this task. (Note: Different recursive calls will print different letters in this word.)


```
typedef struct trienode {
    int isWord;
    struct trienode* next[26];
} trienode;

void printfirst(trienode* root) {

    if (root->isWord) return;

    for (int i=0; i<26; i++) {
        if (root->next[i] == NULL)
            continue;
        // What is true if we get here?
        // A word uses this letter link.
        printf("%c", 'a'+i);
        printfirst(root->next[i]);
        break; // important to force only 1 word print.
    }
}
```

11) (5 pts) Convert 782 in base 10 to base 4.

12) (10 pts) There are n teams in a software competition. Each team has m students. Each student knows some subset of programming languages. There are 20 recognized languages for the competition, numbered 0 to 19, so the subset of languages a single student knows can be stored in a single integer bitmask, which indicates that a student knows language i, if and only if bit i is set to 1. A team can complete a task that requires language i as long as at least one of its team members knows language i. Write a function that takes in an array, languages, where languages[i][j] stores an integer bitmask representing the set of languages that student j on team i knows, as well as the previously mentioned values of n and m, and returns a single integer bitmask storing the set of languages such that all teams could complete a task with that particular language. (Thus, if all teams have at least one individual who knows languages 0, 1, 3 and 5, and the same statement can't be made about any other language, then the function should return $101011_2 = 44$.




```
int getwinmove(int** languages, int n, int m) {
    // Initialize to all languages.
    int res = (1<<20)-1;

    /* Fill in code here. */

    return res;
}
```

```
for (int i = 0; i < n; i++) {
    int team = 0;
    for (int j=0; j<m; j++)
        team |= languages[i][j];
    res &= team;
}
```

} → OR will get all languages covered by team

res = 111 (7)

team = 101 (5)

785=5

101 × 35

100011

13) (3 pts) After which American linguist/computer scientist/philosopher is Chomsky Normal Form named after? (Note: He was born on December 7, 1928 and is still alive!)

_____