COP 3502 Section 2 Exam #1 - Part A (Dynamic Memory Allocation) - Solution

Date: 6/8/2020

Start Time: 4:00 pm EST
End Time: 4:25 pm EST

1) (15 pts) Pascal's Triangle stores the values called binomial coefficients. The triangle has rows, and each subsequent row has 1 more item than the previous row, and each subsequent row can be calculated from the previous row. Specifically, we calculate the next row of the triangle given the previous row as follows:

a) Put 1 in the first and last entry of the new row.
b) For every other entry in the new row, the entry in column i is the sum of the entries in column i-1 and column i of the previous row.

Consider the following example:

Row 5 of the triangle has the following six entries:

1	5	10	10	5	1

To create row 6, we will have 7 entries (1 more than the number of entries in row 5), putting 1 on both ends:

1						1

Now, to calculate the rest of the entries, add 1+5, 5+10, 10+10, 10+5, and 5+1, respectively, yielding:

1	6	15	20	15	6	1

For this question, write a function that takes in an integer array storing a row of Pascal's Triangle (curRow), and the length of that array (length) and does the following:

1. Dynamically allocates a new array of size length + 1.
2. Fills this new array with the appropriate values of the next row of Pascal's Triangle, given that curRow contains the current row of Pascal's Triangle.

3. Frees the array pointed to by curRow.

4. Returns the newly created row of Pascal's Triangle.

The function prototype is provided below:

```
int* getNextRow(int* curRow, int length) {
    // Fill in answer here.
}
```

Note: You may assume that the input array is at least length 1.

Solution

```
int* getNextRow(int* curRow, int length) {

    int* res = malloc((length+1)*sizeof(int));     // 4 pts
    res[0] = 1;                                    // 1 pt
    res[length] = 1;                               // 1 pt

    for (int i=0; i<length-1; i++)                 // 3 pts
        res[i+1] = curRow[i] + curRow[i+1];        // 3 pts

    free(curRow);                                  // 2 pts
    return res;                                    // 1 pt
}
```

2) (10 pts) For the purposes of this question, assume that an integer is stored using 4 bytes, a character is stored using 1 byte and a char* is stored using 4 bytes. The code below dynamically allocates some memory. If the input entered by the user is:

```
6
apple
banana
cantaloupe
dragonfruit
egg
fruit
```

how many **bytes** of memory are **dynamically** allocated by this code segment?

```
int n;
scanf("%d", &n);
int* sizes = malloc(n*sizeof(int));
char** words = malloc(n*sizeof(char*));

for (int i=0; i<n; i++) {
  char temp[100];
  scanf("%s", temp);
  sizes[i] = strlen(temp)+1;
  words[i] = malloc(sizes[i]*sizeof(char));
  strcpy(words[i], temp);
}
```

Note: Your response will be graded as if you don't have access to a calculator. Thus, please show the work you would normally show if you didn't have a calculator. Nearly all of the points for this question are based on the work and logic and not the final answer.

Solution

This line: int* sizes = malloc(n*sizeof(int));
allocates n = 6 times 4 bytes/int = **24 bytes of memory.**

This line: char** words = malloc(n*sizeof(char*));
allocates n = 6 times 4 bytes/ptr = **24 bytes of memory.**

This line: words[i] = malloc(sizes[i]*sizeof(char));
runs six times, and the values of sizes[0], sizes[1], …, sizes[5] are 6, 7, 11, 12, 4 and 6, respectively, 1 more than the length of each of the corresponding words. Since sizeof(char) equals 1, the total # of bytes of dynamically allocated memory here is

6 + 7 + 11 + 12 + 4 + 6 = **46 bytes.**

Thus, the total number of bytes dynamically allocated for this specific example is 24 + 24 + 46 = 94 bytes.

Grading: 2 pts for sizes array,
2 pts for words array,
6 pts for adding up each of the word lengths plus 1.
If they forget the +1 for each word (-2 total)
If they just add wrong (-1 total)
If they count the number of letters in any word incorrectly but it's clear they indicated that they knew what to do (-1 total even if there are multiple incorrect Counts.)