COP 3502 Section 17 Quiz #1 - Part A (Dynamic Memory Allocation) Solution

Date: 5/29/2020

Start Time: 2:30 pm EST
End Time: 2:55 pm EST

1) (10 pts) Write three lines of code in C that do the following: (a), declare an integer variable n, (b) read in from standard input an integer and store it in the variable n, (c) dynamically allocate an array of n floats called array, all set to 0.0. Note, you MUST DO each step, including (c) with a single line of code. (Note: for doubles, if all bits are set to 0, then the value of the number is 0.0 as well.)

Solution

```
int n;
scanf("%d", &n);
float* array = calloc(n, sizeof(float));
```

Grading: 1 pt int n, 2 pts scanf, 1 pt float*, 1 pt array, 1 pt =, 1 pt calloc, 1 pt n for first arg, 1 pt sizeof, 1 pt float. (Note: sizeof(float) is usually 4, so you can give full credit if they put 4 here.)

2) (10 pts) Included below is a section of code allocating memory dynamically for a data structure. Assume that the struct involved (sodacan) does NOT have any dynamically allocated memory. Properly free the memory that is allocated by the code segment.

```
int* ptrN;
ptrN = malloc(sizeof(int));
scanf("%d", ptrN);
sodacan** cans = malloc((*ptrN)*sizeof(sodacan*));

for (int i=0; i<(*ptrN); i++) {
  cans[i] = malloc(sizeof(sodacan));
}
```

Solution

```
for (int i=0; i<(*ptrN); i++)
    free(cans[i]);

free(cans);
free(ptrN);
```

Grading: 3 pts for loop, 3 pts free(cans[i]), 2 pts free(cans), 2 pts free(ptrN). Take off 1 pt if order is invalid. For loop must come before free(cans)…free(ptrN) can be in anywhere in the order.

3) (5 pts) What would be a possible rationale for using realloc on an array to reduce the amount of memory reserved for the array?

Solution

One might not want to reserve too much unused extra memory, so if they see an array is at less than 25% capacity, it may make sense to realloc the array to be half the size. For a single array this isn't a big deal, but imagine a program with a 1,000,000 arrays, most of which were very small but a few were somewhat large. If each of these 1,000,000 arrays were largely unused, a program could run out of memory or run much more slowly due to hogging up necessary memory.

Grading: Give full credit to any response that clearly indicates the idea of not wanting to reserve large amounts of unused memory. If the student probably has the right idea but is expressing it poorly, give partial credit.