COP 3502 Section 16 Quiz #1 - Part A (Dynamic Memory Allocation) Solutions

Date: 5/29/2020

Start Time: 1:30 pm EST
End Time: 1:55 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 doubles 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);
double* array = calloc(n, sizeof(double));
```

Grading: 1 pt int n, 2 pts scanf, 1 pt double*, 1 pt array, 1 pt =, 1 pt calloc, 1 pt n for first arg, 1 pt sizeof, 1 pt double. (Note: sizeof(double) is usually 8, so you can give full credit if they put 8 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 n;
scanf("%d", &n);
int* sizes = malloc(n*sizeof(int));
sodacan** cans = malloc(n*sizeof(sodacan*));

for (int i=0; i<n; i++) {
  scanf("%d", &sizes[i]);
  cans[i] = malloc(sizes[i]*sizeof(sodacan));
}
```

Solution

```
for (int i=0; i<n; i++)
    free(cans[i]);
free(cans);
free(sizes);
```

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

3) (5 pts) In the class example cs1list.c, we created a list that dynamically grew and shrank as necessary. What was the rationale for doubling the list size instead of adding 1 when the list filled up?

Solution

Reallocating the list is potentially time consuming, if all the existing memory has to be copied from one place to another. If we only add 1 to the existing list size, we might have to reallocate for EVERY add operation after the list initially fills. By doubling the list size, we ensure that after one realloc, another one doesn't get called for a good amount of time.

Grading: Give partial credit if the student has the right idea but is expressing it poorly. Give them full credit if they are clear that too many reallocs could waste time. If there is an alternate answer that you think deserves merit, feel free to give it some partial credit.