Midterm 1

Tuesday, March 30, 2021 10:21 AM

"I pledge my honor I have abided by the Stevens Honor system."
- Alex Jaskins

1. [5 points] What are the benefits of creating data structures using the heap over using contiguous memory locations in the application space? Provide at least one concrete example to justify your answer. Your argument should exemplify a tangible benefit.

Instead of creating new names and references in the app space, using the HEAP allows the program to run faster by not having to grab a enough memory to run beforehand. The program does not have to worry about how much memory space is required for the program, so each individual function can use enough memory easily and when the function is done it is not eating into the rest of the memory being used by the program while it is still running.

An example of this is using Google Chrome. If it were not using the HEAP when opening a new tab, the program would have to prepare for a certain amount of tabs to be opened, which

of tabs to be opened, which wastes memory, because the program would always be using enough memory for 20 (example) tabs to be opened even if the user only opens 1 or 2 tabs. Using the HEAP would eliminate this issue by only grabbing memory when a tab is opened and can continue adding memory (until no more can be used by the computer).

2. [15 points] Prove, or disprove, the following:

(a) $3n^2 + 4n^3 + 5 = O(n^2)$ False. Worse case scenario $= O(n^3)$

(b) $8\log n + 4n = O(n)$ True. $4n$ (worse case scenario) is dominant

(c) $2^n + 4n^3 = O(n^3)$ False. 2^n is worse than polynomial $= O(2^n)$

(d) $2n^2 + 3n = \theta(n^2)$ True. n^2 is more dominant than n

(e) $2^n + n^2 + O(n) = O(n^2)$ False. 2^n is worse than polynomial $= O(2^n)$

3. [10 points] The Fibonnaci series looks like the following F = 0,1,1,2,3,5,8,⋯, where the n-th term is the sum of *(n - 1)* th and *(n - 2)* th term. Present a **recursive** function that prints the first *n*-terms of the sequence. Implement `int getNthTermFibonnaci(int n)`

```cpp
#include <iostream>
using namespace std;

int getNthTermFibonnaci(int n)
{
    if (n <= 1)
    {
        return n;
    }
    else
    {
        return getNthTermFibonnaci(n - 1) + getNthTermFibonnaci(n - 2);
    }
}

int main()
{
    int num;
    int i = 0;
    cout << "How many terms do you want the Fibonnaci Series to go to?" << endl;
    cin >> num;
    cout << " Fibonnaci Series: " << endl;
    while (i < num)
    {
        cout << " " << getNthTermFibonnaci(i);
        i++;
    }
    return 0;
}
```

```
PS C:\Users\agaskins\Desktop> g++ .\fibonacciRecursion.cpp
PS C:\Users\agaskins\Desktop> ./a.exe
How many terms do you want the Fibonnaci Series to go to?
20
 Fibonnaci Series:
 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
PS C:\Users\agaskins\Desktop>
```

4. [10 points] Why are linked-lists better than arrays? Do list at least three advantages to a linked-list structure over static arrays?

- To insert a value at some point in an array, the whole array must be shifted to accomodate, whereas it only requires two pointers to do so in a Linked-List. Thus, making it easier to add and remove elements.

- Linked-Lists can be added to indefinitely, whereas arrays eventually get filled

Linked Lists ... whereas arrays eventually get filled or have to be resized.

- Unlike Linked-Lists, when elements are removed from an array it leaves empty space that wastes comuter memory.

- Linked-Lists are also faster than arrays

5. [5 points] What is the difference between a "const" variable and a "static" variable? When do you use each one of these in your program?

A static variable gets created only once and is shared across different function calls used throughout the entire program. It is often useful with recursive functions to keep up with the copies that are created of themselves.

A const varible sets the value of a variable to be certain and promises it can't be changed. They are typically used in individual functions but can be used anywhere in the program to initialize a variable that can't be changed. (value is constant)

6. [5 points] What is the "signature" of a function?

The signature of a function is what the function returns, the name of the function and its arguments.

function [illegible], the name of [illegible]
and its arguments.