Hello class,

Here is a set of sample problems that you can use while studying for your final examination. The problems are in order of topic or randomized. I strongly recommend you look at these after you feel like you have prepared well enough for the exam. I also strongly urge you to look at the randomized problems over the ones ordered by topic. On an exam, the order or problems will be random with respect to topic (and sometimes random with respect to difficulty). Solving problems in random order will be far better practice than doing it in order of topic. I will not be providing solutions (because quite honestly I don't have any and making this problem set was exhausting enough). I am happy to answer any question you have on these problems and encourage you to discuss these amongst yourselves and on Piazza. I have no anticipation of anyone solving all 100+ problems but clearly the more you solve the better off you will be on the exam. I've indicated problems that I think are really challenging on the set so that you don't sink too much time into them.

All the best and happy studying!

Carmen

Part 1

1. Use the Euclidean algorithm to find the gcd of 1350 and 2275
2. Write down a non-recursive and recursive algorithm to compute the gcd of two numbers.
3. What does the following code print?

```
#include <stdio.h>
int main(void) {
  int a=1, b=2, c=3, d=4;
  printf("%d %d ", ++a, b++);
  printf("%d %d ", 3*d/3, 3*(d/3));
  printf("%d %d ", 1+2-3*4/5%6+7, );
  printf("%d %d %d %d ", -14%5, -14%(-5), 14%5. 14%(-5));
  return 0;
}
```

4. Write a while loop to sum the first n even natural numbers (start with 0) where n is a number that is given as input from the user. Give an error message if $n < 0$.
5. Convert the following three numbers to binary: 14, 83, 99
6. Convert the following three numbers to decimal: 01001111, 11001100, 01110111
7. Explain the difference between little endian and big endian byte order.
8. Describe overflow errors. Give a concrete example of unexpected behaviour that can occur if not accounting for overflow issues.
9. What does the following code print?

```
#include <stdio.h>
#include <stdbool.h>
int main(void) {
  int a=1, b=2, c=3, d=4, e=0;
  printf("%d %d ", a+=2, b-=3);
  printf("%d %d ", true && (false || true), !true || false);
  printf("%d %d ", 2 <  5 <  3, e != 0 && 4/e > 2);
  bool P = true, Q = false;
  printf("%d %d %d %d ", !(P||Q), ~(P&&Q);
  return 0;
}
```

10. What does the following code print?

```
#include <stdio.h>
#include <stdbool.h>
int main(void) {
  char a=1, b=2, c=3, d=4, e=0;
  printf("%d %d %d %d %d %d", a&b, c|d, a^c, a<<1, d >> 1, ~b);
  return 0;
}
```

11. What does the following code print?

```c
#include <stdio.h>
int main(void) {
  int n = 20;
  if(n < 100){
    if (n > 10){
      printf("Fun!\n");
    } else{
      printf("Bananas!\n");
    }
  else if (n < 200){
    printf("Weird\n");
  }
  if (n < 300) printf("Finally!\n");
  return 0;
}
```

12. Write a C function that takes in a number and sums all of the digits in the tens, thousands, hundred thousands, and ten millions positions (that is, every other position beginning with the tens position).

13. Write a C function that reverses a number.

14. Write a C function that counts the number of ones a decimal number has in binary.

15. Write a program that takes in an integer from the user and if the number is divisible by 3 we print "zig", if the number is divisible by 7 we print "zag" and if the number is divisible by 21, we print "zigzag".

Part 2

1. What does the following code do?

```c
#include <stdio.h>
int main(void) {
  int n, a, sum=0;
  scanf("%d",&n);
  for(int i=0; i<n; i++){
    scanf("%d",&a);
    if (a < 0) continue;
    if (a == 0) break;
    sum += a*a;
  }
  printf("%d\n",sum);
  return 0;
}
```

2. The following code contains an error - find it and suggest a correction

```c
#include <stdio.h>
int main(void) {
  int a[10] = {0,1,2,3,4,5,6,7,8,9};
```

```
  int sum=0;
  for(int i=0; i<=10; i++) sum += a[i];
  printf("%d\n",sum);
  return 0;
}
```

3. The following code contains an error - find it and suggest a correction

```
#include <stdio.h>
int main(void) {
  int a[5] = {0,1,2,3,4,5};
  int sum=0;
  for(int i=0; i<sizeof(a)/sizeof(a[0]); i++) sum += a[i];
  printf("%d\n",sum);
  return 0;
}
```

4. What does the following code do?

```
#include <stdio.h>
int main(void) {
  int a[4][4] = {{1,2,3,4},{5,6,7,8},{2,4,6,8},{3,5,7,11}};
  int sum=0, sign=-1;
  for(int i=0; i<sizeof(a)/sizeof(a[0]); i++)
    sum += (sign *= sign) * a[0][i];
  printf("%d\n",sum);
  return 0;
}
```

5. Write a function that computes the maximum of three integers.
6. Write a function that computes the gcd of three integers, defined as $\gcd(a, b, c) = \gcd(\gcd(a, b), c)$.
7. Write a function that computes the number of days until the next Christmas (December 25th). Your function should take in three integers, a **current year**, a **current month** and a **current day**. Don't forget about leap years!
8. Implement the Sieve of Eratosthenes.
9. Create a header, implementation and main function that contains functions relating to regular polygons. You should implement the functions `perimeter`, `area`, `interior_angle`. Each function should take in the number of sides of your regular polygon and the length of one side. (Hint: Look up what an apothem is for help with the area computation).
10. Write a program that computes the lcm of two integers recursively.
11. Write a recursive program that calculates the number of ways that a person can travel on a rectangular grid by starting on the bottom left and moving to the top right only moving up and right. What if you allow for a finite number of left or down moves?
12. Write a program that reads a list of integers from standard input, one per line, and outputs them in reverse order. You may assume that the last integer in the list is 0, and 0 does not appear anywhere else in the list. Do not use arrays. You must use recursion to solve this problem.

13. *Challenge* Write a function that takes in an array of size `n` and an integer `k` such that $0 \leq k \leq n-1$ and all elements of the array are between 0 and k and return the most frequent element in the array. You may assume that this element is unique. You can assume you will not have any overflow errors with your algorithm. You are only allowed to use a constant amount of additional space (that is, you cannot initialize a variable number of variables [say for example depending on n or k]). We sometimes say this is using $O(1)$ space. You may mutate the array but must return it to its original state by the end of the function.

Part 3

1. Write a function that takes in a day, month and a four digit year and outputs the date in the format `dd/mm/yyyy`.
2. Describe what `%±m.pX` does where `X` is one of `d`, `e`, `f` and `g`. If you've covered this, discuss this with `X` as `c` as well.
3. Write a function that takes in an integer and prints it out as an 8 digit integer, left padding with zeroes as needed.
4. Give a brief and informal explanation of how the IEEE 754 floating point standard works. You needn't be exact but should have the general components (that is, sign, fraction, exponent).
5. Convert the decimals $0.1, 0.3, 0.5, 0.7, 0.9$ into a binary decimal.
6. Convert the binary decimal $0.1\overline{0010}$ into a decimal number. You may need to use some knowledge of infinite geometric series from high school or calculus to help.
7. What are the relative and absolute errors of using the approximation 2 for $\sqrt{3}$? What about using 3 for $\sqrt{8}$?
8. Explain why the following code prints `Not cool` on my machine. How should we be properly testing for equality of floating point numbers in C?

```c
#include <stdio.h>
int main(void) {
  double a = 7.0/12.0;
  double b = 1.0/3.0;
  double c = 1.0/4.0;
  if (b+c==a) printf("Everything is Awesome!");
  else printf("Not cool",b+c-a);
}
```

9. What is the value of `a` if `double a = 1/3` is executed?
10. Write a program that takes in the hypotenuse and one of the two non $\pi/2$ angles in a right angle triangle and computes all the side lengths correct to at least 4 decimal places. Extend this by making a header, implementation and main file with this information and also implement the area function.
11. Write $3x^3+4x^2+9x+2$ using Horner's method. What advantages are there in using Horner's method to compute the value of a polynomial versus other methods?
12. Implement any root finding algorithm.
13. Describe the difference in the bisection method between using `fabs(f(m)) < EPSILON` and using `fabs(a-m) < EPSILON`. Which is guaranteed to give a bound on the root?

14. Write a function that takes in a double `x` and a function pointer `f` (with a single double parameter and a single return value of a double) and returns the value of `f(x+1) - f(x)`.

Part 4

1. Write a `struct` for a 2-D Video Game Character. The character should have a (x, y) position, an id number (an `int` that you should assert to be greater than 0) and a level. You should also write functions `fight` which takes in two characters and returns the id number of the one with the higher level (or -1 if there is a tie). Also, include four functions `move_up move_down move_left move_right` which changes the (x, y) position by one unit and returns the character with this valued changed. (Note: If you have seen pointers, implement these functions with pointers instead of character copies).

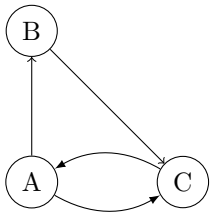
2. Recall our time of day struct:

```
struct tod{ int hours, minutes;}
```

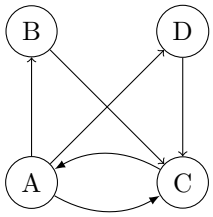
   Implement the comparison operator which takes in two `struct tod` parameters and return -1 if the first is less than the second, 1 is the first is greater than the second and 0 if the two times are equal.

3. Write down the formulas for the non-randomized and the randomized page rank algorithms.

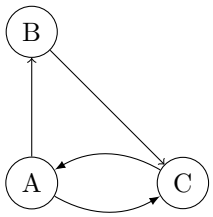
4. Compute the non-randomized page rank algorithm of the following



5. Compute the non-randomized page rank algorithm of the following



6. Compute the randomized page rank algorithm of the following (use $\delta = 0.5$ and if you want a computational challenge, try it with $\delta = 0.8$).



7. Write code for the page rank algorithm.

Part 5

1. What does the following code print?

```c
#include <stdio.h>
int main(void) {
  int a[10] = {10,11,12,13,14,15,16,17,18,19};
  int *p = a;
  int *q = a+5;
  printf("%d\n", *(p+3));
  printf("%d\n", *(q+3));
  printf("%p\n", p); //Exact value of this is hard.
  printf("%p\n", p+1); //Give answer relative to line above.
  printf("%p\n", q-2); //Give answer relative to the above.
  printf("%d\n", p[2] + 3[p]);
  printf("%d\n", p[q[2]-11]);
  printf("%d\n", *(q++));
  printf("%d\n", *(++q));
  printf("%d\n", ++(*q));
  printf("%d\n", q-p);
  return 0;
}
```

2. What does the following code print? Further, write a function that takes in a pointer to a `date` and an integer `n` and returns what day it is in the future. Don't forget about leap years!

```c
#include <stdio.h>
typedef struct{
  int day, month, year;
} date;
int main(void) {
  date *past = malloc(sizeof(date));
  past->day = 17;
  past->month = 10;
  past->year = 2009;
  printf("%d", past->year % 4);
  return 0;
}
```

3. What does the following code print?

```c
#include <stdio.h>
#include <stdlib.h>
int *foo(int a[], int n){
  int *b = malloc(n*sizeof(int))
  for(int *p=a; p<a+n; p++){
    if(*p %2==0){
      *b = -*(p + (p-a));
   }else{
      *b = *(p + (p-a));
    }
}
```

```
int main(void) {
  int a[10] = {0,1,2,3,4,5,6,7,8,9};
  int *b = foo(a,10);
  for(int *p = b; p < b+10; p++){
    printf("%d\n",*p);
  }
  free(b)
  return 0;
}
```

4. Write a function that adds the values of an array together. Use pointer arithmetic.
5. Write a function that swaps two integers. The integers should be passed to your function by reference (so that the actual values in the calling function are modified by the end of your function).
6. Explain the difference between the stack and the heap.
7. How do we prevent memory leaks when allocating to the heap?
8. Write a small function that takes in a integer pointer and prints out `Null` if the pointer is the null pointer and prints out `Not Null` otherwise.
9. Implement a `struct` that models a classroom. A classroom consists of a course number, a `struct tod` of the current time of day, an integer representing the number of students and a class list that consists of student ID numbers (integers between 0 and 10 million). (If you've seen strings already, you can use names instead of ID numbers). Use the struct hack to help implement this. Further, create a function that takes in a pointer to a `struct class` and modifies the time of day. Lastly, create functions `createClass` and `deleteClass` which allocates and frees memory.
10. Write the `vectorSet` function from the vector class.

Part 6

1. What does the following code print?

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(void) {
  char *a = malloc(20*sizeof(char));
  a[0] = 'c';
  a[1] = 'a';
  a[2] = 'r';
  a[3] = '\0';
  strcat(a," race");
  printf("%d", strlen(a));
  printf("%s", a);
  strcpy(a,"fee");
  strncpy(a,"line segment",3);
  strncpy(a,"g");
  printf("%s", a);
  memset(a,'\0',10);
  a[0] = 99;
```

```
    a[1] = 97;
    a[2] = 's'-1;
    a[3] = 97+4;
    printf("%s", a);
    return 0;
}
```

2. List some of the differences between using `const` and macros.
3. What are the ASCII encodings for the letters 'A' and 'a'? What is their difference and what does this imply about the gap between upper and lower case letters?
4. What is the UTF-8 encoding of $U + 04BF$? You may use the table from class.
5. What is the UTF-8 encoding of $U + 234BF$? You may use the table from class.
6. What is the UTF-8 encoding of $U + A4BF$? You may use the table from class.
7. Write a C function that takes in a character and checks to see if it is a capital or lower case letter.
8. Write a code that swaps the case of every letter. You may assume that the input is a nonempty string of all Latin alphabet letters except for possible single white spaces.
9. Write a function that takes in an integer and outputs the corresponding UTF-8 Unicode encoding of the character represented by that integer. You may assume that the integer has at most 21 bits of nonzero entries and that any bits left of the right most 21 bits are 0 (so you can actually do the encoding).
10. Write a function that compares two Latin alphabet strings of capital letters as though "C" was the first letter of the alphabet (and Z, A, B are the last three letters). For example, `CART` would come before `BACON` in this sorting. Your function should return -1 if the first word is less than the second word, +1 if the first word is bigger than the second word and 0 otherwise.
11. Write a function `is_substring` which takes in two strings `s` and `t` and returns true if and only if the string `s` is completely contained inside the string `t` and false otherwise. You can assume `s` and `t` are non-null. Note that the empty string is a substring of all strings. For example, the string `RAN` is a substring of `CURRANT` but not of `RAMPANT`.
12. Write a C function that counts the number of numerals (digits between 0 and 9) in a string.

Part 7

1. Write the definition of $O(f(n))$.
2. What does $g(n) = O(f(n))$ mean?
3. Determine using the definition directly if $O(n^2 + n) = O(2n + 0.5n^2)$.
4. Rank the following functions in order of growth from slowest to largest

$$\log n \qquad n^{10} \qquad n^{0.1} \qquad n \log n \qquad 100n! \qquad 42^n \qquad n^n \qquad \frac{1}{n} \qquad 1$$

5. Write a linear searching algorithm. What is it's runtime in the best case? What about the worst case? Explain what an average case might look like and identify the runtime in this case.

6. Prove that $n \log n = O(n^2)$. Do this using theorems from the course and using the definition directly.

7. *Challenge* Give an example of two functions $f(n)$ and $g(n)$ such that $f(n) \notin O(g(n))$ and $g(n) \notin O(f(n))$. Hint: This is easier to do with non-continuous functions.

8. Prove that if $g_0(x) = O(f_0(x))$ and $g_1(x) = O(f_1(x))$ then $g_0(x)g_1(x) = O(f_0(x)f_1(x))$.

9. Give an example of two non-equal functions $f(n)$ and $g(n)$ such that $O(f(n)) = O(g(n))$. Prove your answer is correct.

10. Given the array `int a[] = {19,3,7,2,11,47,35,13,10}`, run the selection sort algorithm on this input and describe the array after each outer loop iteration.

11. Given the array `int a[] = {19,3,7,2,11,47,35,13,10}`, run the insertion sort algorithm on this input and describe the array after each outer loop iteration.

12. Explain the best, worst and average case runtimes of both selection sort and insertion sort. Which of these would you use in practice and why?

13. Write an $O(n)$ function that takes an integer array and returns another array where the ith entry is the product of all other entries except at i. You may not use division and you may only allocate $O(n)$ space (that is, you may allocate some constant times n amount of extra variable space). Can you solve this with only one additional array? The original array should remain in its initial state.

14. Write a function that takes in an array returns the subarray with the largest sum. A subarray is a consecutive array of elements contained in the original elements. The array might have negative, positive and zero entries. How quick can you make this code run? *Challenge* Can you make this run in $O(n)$ time? You should at least be able to reduce this to $O(n^2)$.

15. Write a function that takes in an array returns the subarray with the largest product. A subarray is a consecutive array of elements contained in the original elements. The array might have negative, positive and zero entries. How quick can you make this code run? *Challenge* Can you make this run in $O(n)$ time? You should at least be able to reduce this to $O(n^2)$.

16. *Very Hard Challenge* Given a string, determine the longest palindromic subsequence, that is, the largest subsequence (contained sequence of letters in order) of the string that is also a palindrome. Can you get the runtime all the way down to $O(n)$? What if we change subsequence to substring (so the letters must also be consecutive)?

17. What is the runtime of the following code?

```
int alpha(char *s){
  char *t = s;
  char *u = malloc((strlen(s)+1)*sizeof(char));
  for(;*t;t++){
    strcpy(s,t);
  }
  return -1;
}
```

18. What is the runtime of the following code?

```
int beta(int n){
  int s=0;
  for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
      s++;
    }
  }
  return s;
}
```

19. What is the runtime of the following code?

```
int gamma(int n){
  int s=0;
  for(int i=0; i<1000000; i++){
    for(int j=0; j<n/10; j++){
      s++;
    }
  }
  return s;
}
```

20. What is the runtime of the following code?

```
int epsilon(int n){
  int s=0;
  for(int i=0; i<n; i++){
    for(int j=i; j<2*n; j++){
        for(int k=i; k<3*n; k++){
         s++;
       }
    }
  }
  return s;
}
```

21. What is the runtime of the following code?

```
int zeta(int a[], int n){
  int s=0;
  for(int i=1; i<n; i*=2){
    if (a[i] ==0) return -1;
    else{
      s++;
    }
  }
  return s;
}
```

22. What is the runtime of the following code?

```
int iota(int n){
  int s=0;
  for(int i=1; i<n; i+=2){
   for(int j=i; j<n*n; j++){
     s++;
   }
  }
  return s;
}
```

23. What is the runtime of the following code?

```
int delta(int n, int a[n][n] ){
  int s=1,i=0;
  while(s < n){
    if (a[0][i] < 0){
     s *=2;
    }else{
     s += 2;
    }
  }
  return s;
}
```

Part 8

1. Given the array `int a[] = {19,3,7,2,11,47,35,13,10}`, run the merge sort algorithm on this input and describe the array after each outer loop iteration.
2. Given the array `int a[] = {19,3,7,2,11,47,35,13,10}`, run the quick sort algorithm on this input and describe the array after each outer loop iteration.
3. Explain the best, worst and average case runtimes of both merge sort and quick sort. Which of these would you use in practice and why?
4. Quick sort in the worst case is worse than many other sorting algorithms such as merge sort. Why do many companies still choose to use quick sort (or slight variations of it)?
5. Code the binary searching algorithm.
6. Estimate the number of comparisons on a 10^7 array of sorted elements using binary searching.
7. Run through our binary searching algorithm on a sorted array of odd numbers less than 1000 recording which elements are probed.
8. Explain tail call elimination.
9. If an array was almost sorted, which algorithm would you use to sort it any why?
10. Rewrite the binary searching algorithm so that it is recursive instead of iterative.