Instructions. Please show work where needed. Answers without appropriate justification will receive no credit. Each of the following 20 problems is worth 5 points. When asked to write code, you may declare local variables if you think they are needed.

1. Suppose that \( i \), \( j \) and \( k \) are variables of type `int` having values 10, 20 and 3 respectively. Find the values of the following expressions.

   (a) \( i + j + k / k \) = 

   (b) \( (i + j + k) / k \) = 

   (c) \( i / j / k \) = 

   (d) \( j \% i - j / i \) = 

   (e) \( i + j * k / i * (-4) \) = 

2. Write `if` (or `if-else`) type decision structures for each of the following. Assume that \( i \), \( j \) and \( k \) are variables of type `int`.

   (a) Assign \( i \) the largest of the values of \( i \), \( j \) and \( k \).

   (b) Assign \( i \) the value 0 if \( i \) is an odd number greater than 1000.

   (c) Increment \( i \) by 1 if the value of \( j \) is between 0 and 5 inclusive, increment \( i \) by 2 if \( j \) is between 5 and 25 inclusive and increment \( i \) by 3 if \( j \) is between 26 and 100 inclusive.
3. Suppose that $P$, $Q$, and $R$ represent conditions such that $P = \text{false}$, $Q = \text{true}$ and $R = \text{true}$. Are the following true or false?

(a) $Q \land P \land R$

(b) $!(P \land Q) = !P \land !Q$

(c) $(P \land Q \land !R) \land (-1 >= 4)$

(d) $(P \land (Q \land R)) = ((P \land Q) \land (P \land R))$

4. Consider the following block of unindented code.

```plaintext
if (i >= k) {if (a < b) if (c > d) x = 1; else x = 2; else if (a > e) x = 3; else x = 5;

Rewrite this code using indenting so that it is obvious which else goes with which if.
5. Fill in the blank so that each loop prints exactly 12 asterisks.

(a) 
```cpp
for (int i = ______; i <= 12; i++)
    cout << "***";
```

(b) 
```cpp
int i = 5;
while (i > ______) {
    cout << "***";
    i--;
}
```

(c) 
```cpp
int i = ______;
do {
    cout << "***";
    i = i + 2;
} while (i < 30);
```

6. For each of the following loops, how many times is the asterisk printed. If there is an in.nite loop, write ‘INFINITE LOOP’.

(a) 
```cpp
int i = 0, j;
while (i <= 10) {
    j = 0;
    while (j < i) {
        cout << "***";
        j++;
    }
    i++;
}
```

The asterisk is printed ______ times.

(b) 
```cpp
for (int i = 0; i < 6; i++)
    for (int k = 3; k < 7; k++)
        for (int m = 0; m < 9; m++)
            cout << "***" << endl;
```

The asterisk is printed ______ times.

(c) 
```cpp
int i = 1;
do {
    cout << "***";
    i--;
} while (i < 1);
```

The asterisk is printed ______ times.
7. Write a function called `curveGrade` defined as follows. The function `curveGrade` has exactly one reference parameter `g` of type `float` that represents a student's grade. If `g > 70.0`, then the curve is 5% (That is, the curved grade is 5% more than the old grade.) If `40.0 < g <= 70.0`, the curve is 8% and finally, if `g <= 40.0`, the curve is 10%.

```c
void curveGrade (float& g) {
```

8. Write a function called `extract` that extracts the substring of string `str` from indices `i` to `j` of `str` and returns it in the string `substr`. For example, if `str = LOYOLA`, `i = 3` and `j = 5`, then `substr = CLAn0`. You may assume that `i < j` and that `str[j]` is not the null character.

```c
void extract (char str[], int i, int j, char substr[]) {
```
9. Consider a class called `dice` defined below.

```cpp
class dice {
private:
    int getRandNum() { //returns a random number
        return rand() % 6 + 1;
    }

    int face1, //number on face 1
        face2; //number on face 2

public:
    static void**********;
};
```

(a) Write a constructor that creates an object of type `dice` and initializes `face1` to the number `m` and `face2` to the number `n` where `1 <= m, n <= 6`.

(b) Write a member function called `getFace1` that returns the number on face 1 of the dice.

```cpp
int getFace1() {
```

(c) Write a member function called `getSum` that returns the sum of the numbers on the faces of the dice.

```cpp
int getSum() {
```
10. For this problem, use the dice class of problem 9.

(a) Write a member function called check that returns 1 if the sum of the numbers on the faces is equal to $n$ where $2 \leq n \leq 12$. If the numbers do not add up to $n$, the function should return 0.

```c
int check (int n) {
```

(b) Write a member function called roll that rolls the dice. Use the private function getRandNum.

```c
void roll () {
```

(c) Write a member function called printDice that prints out the numbers on the faces in a readable way.

```c
void printDice () {
```
11. For this problem, use the dice class of problem 9. Write a complete C++ program that will create and roll a pair of dice n times and count the number of times the sum of the numbers on the faces add up to 7 or 11. Your program should prompt the user for the value of n, perform the computation, print out the result and terminate. The first couple of lines of the program are given below. Use member functions that your wrote in the previous two problems!

```cpp
#include <stdlib.h>
#include <iomanip.h>
#include <iostream.h>
#include "dice.h" //header file

void main () {
    dice d = dice(1,1); //a pair of dice
    unsigned int seed;
    int n, count; //count holds the number of times
                  //the dice come up 7 or 11
    cout << "Enter a seed for the ";
    << "random number generator: ";
    cin >> seed;
    srand(seed); //start random number generator

    // Your code here
}
```
12. Do the following base conversions.

(a) In binary, the number 41 = _________________

(b) In base 3, the number 54 = _________________

(c) In base 16, the number 30 = _________________

13. Find the g.c.d. of the numbers 630 and 154 using function Euclidean. To start the algorithm, let dividend = 154 and divisor = 630. Fill in the values of the variables for the first iteration, and then write down the values for as many other iterations as you need to complete the algorithm. Make sure you specify what the g.c.d of the two numbers are at the end of your computation!

Iteration #1: 

remainder = 

dividend = 

divisor = 

14. Write a recursive function called reverse that will reverse the digits in the positive integer parameter n. For example, if n = 57384, then reverse should print 48375.

    void reverse (int n) {

15. Consider the following recursive function called mystery.

    double mystery (double x, int n) {
        double temp;
        if (n == 0)
            return 1;
        else if (n % 2 == 0) {
            temp = mystery (x, n / 2);
            return temp * temp;
        }
        else
            return x * mystery (x, n - 1);
    }

(a) In exactly one sentence, describe what this function is doing.

(b) Find the value of mystery(2, 5).
16. Consider the following array of numbers called a.

\[
\begin{array}{cccccccc}
2 & 7 & 3 & 0 & 5 & 1 & 8 & 9 & 4 & 6 \\
\end{array}
\]

(a) What happens to the array a after the following segment of code is executed on it?

```cpp
for (int i = 0; i <= 8; i++)
    a[i] = a[i + 1];
```

(b) What happens to the array a after the following segment of code is executed on it? (Use the original array a.)

```cpp
for (int i = 0; i <= 8; i++)
    a[i + 1] = a[i];
```

(c) What happens to the array a after the following segment of code is executed on it? (Use the original array a.)

```cpp
int t;
for (int i = 0; i <= 3; i++) {
    t = a[i];
    a[i] = a[9 - i];
    a[9 - i] = t;
}
```

17. Write a function that is passed two arrays a and b and returns the number of entries in a that are less than the corresponding entry in b. For example, if a = \{3, -1, 7, 4, 11\} and b = \{0, 0, 5, 5\}, then the function would return the number 2.

```cpp
int lessThan (int a[], int b[], int size) {
```
18. Perform four passes of selection sort on the array below.

\[\begin{array}{cccccccccc}
16 & 33 & 56 & 11 & 50 & 31 & 7 & 49 & 60 & 2 \\
\end{array}\]

pass #1: 

pass #2: 

pass #3: 

pass #4: 

19. Perform six passes of insertion sort on the array below.

\[\begin{array}{cccccccccc}
54 & 37 & 17 & 32 & 19 & 52 & 28 & 70 & 81 & 23 \\
\end{array}\]

pass #1: 

pass #2: 

pass #3: 

pass #4: 

pass #5: 

pass #6:
20. Perform function `split` (the function called by `quickSort`) on the array below. The empty arrays below can be used as work arrays

| 0 | 1 | -3 | 6 | -2 | -1 | 2 | 4 | -5 | 6 |

```
after split:          
```

The final value of `pivotLoc` is ____________.