1. [10 points] For each of the following pieces of code, what would be printed to the screen?

(a) int x = 3, y = 9;
    while (x < y)
    {
        cout << x << " " << y << endl;
        x = x + 3;
    }

(b) int x = 3, y = 9;
    do {
        cout << x << " " << y << endl;
        x = x + 3;
    } while (x < y);

(c) int main()
{
    int x = 7, y = 2;

    fn1(x, y);
    cout << x << ", " << y << endl;

    fn2(x, y);
    cout << x << ", " << y << endl;

    return 0;
}

void fn1(int& n1, int& n2)
{
    if (n1 > n2)
        n2 = n2 * 2;
}

void fn2(int n1, int n2)
{
    if (n1 > n2)
        n2 = n2 * 2;
}
2. [5 points] What does this function Mystery do (in general)? Choose one that applies.

```c
int Mystery(int a, int n) {
    int i = 1, x = 0;
    while (i < n) {
        x += a;
        i++;
    }
    return x;
}
```

Hint: Try some example values such as a=2 and n=3 to develop your intuition.

(a) It computes a + n and returns the value.
(b) It computes a * n and returns the value.
(c) It computes a + (n-1) and returns the value.
(d) It computes a * (n-1) and returns the value.
(e) It computes (a-1) + n and returns the value.
(f) It computes (a-1) * n and returns the value.
(g) It computes (a-1) + (n-1) and returns the value.
(h) It computes (a-1) * (n-1) and returns the value.

3. [15 points] Write a function `computeBill` which computes a service bill for a customer at an internet provider based on his/her package and usage hours. This provider has three different subscription packages for its customers:

- Package A: For $9.95 per month, 10 hours of access are provided. Additional hours are $2.00 per hour.
- Package B: For $14.95 per month, 20 hours of access are provided. Additional hours are $1.00 per hour.
- Package C: For $19.95 per month, unlimited access is provided.

Here are some calculation examples for your reference.

- Package A, 12.6 hours => $15.15; 8.6 hours => $9.95
- Package B, 9.2 hours => $14.95; 30.1 hours => $25.05
- Package C, 8.6 hours => $19.95; 40.6 hours => $19.95

The function should have two parameters: a `char` for the package (such as 'A', 'B'), and a `double` for the hours. The prototype is:

```c
double computeBill(char pack, double hours);
```

Note that you can make the package 'C' as the default case. Also note the function should NOT PRINT anything at all.
4. [5 points] Using the function `computeBill` from the previous question, complete the following program.

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

// prototypes
double computeBill(char pack, double hours);

int main()
{
    char package;
    double usage;

    cout << "What is your package (A/B/C)? ";
    cin >> package;
    cout << "How many hours did you use? ";
    cin >> usage;

    // TO DO: Call the function to obtain the bill, and
    // display the bill amount in the form "$xx.yy" (i.e.,
    // a dollar sign and two decimal digits).
```

5. [10 points] Write a function `sumOddInRange`. This function has two int parameters ('begin' and 'end' in the prototype below) which are intended to specify a range in the number line, and returns the summation of all odd integers between the range (both ends inclusive). For example, if the function is called with 3 (for 'begin') and 20 (for 'end'), the function returns the result of 3 + 5 + 7 + ... + 17 + 19.

```cpp
int sumOddInRange(int begin, int end);
```

6. [10 points] Write a function `YesOrNo` whose prototype is

```cpp
bool YesOrNo(const char question[]);
```

This function receives a cstring ('question') and does the following in order:

- Displays the parameter cstring, followed by "'(1 = Yes, 2 = No)'", to cout.
- Receives the user's input (an integer) from keyboard.
- Goes back to the first step when the entered integer was neither 1 nor 2.
- (after a correct value (1 or 2) is obtained) Returns true if the value was 1, or false if the value was 2.

For example, for a function call:

```cpp
bool answer = YesOrNo("Do you like ice cream");
```

A sample session is:
Do you like ice cream (1 = Yes, 2 = No) 3
Do you like ice cream (1 = Yes, 2 = No) -1
Do you like ice cream (1 = Yes, 2 = No) 1

And the function returns true in this case

7. [10 points] Write a function indexOf. This function receives a cstring and a character which is to be searched in the string. This function function returns the index of the search character if it exists in the array, or -1 if it does not. The returned index is 0-based. Also if the array contains more than one occurrence of the character, it returns the index of the left-most occurrence. For instance, if the cstring was "abab" and the search character was 'b', the function returns 1; whereas if the character was 'c', the function returns -1.

Write the function WITHOUT using any C++ library functions.

```c
int indexOf(const char str[], char ch);
```

8. [7 points] For each statement below, say true or false. You do not have to give a reason.

(a) The member variables of a class (all of them) must be of the same type.
(b) The member variables of a class (all of them) must be of C++ primitive type(s) (e.g. int, char, double).
(c) A class can have more than one constructor.
(d) A default constructor can have up to two parameters.
(e) A member function of a class can change the value of a private member variable of the class.

9. [3 points] Consider the following class definition:

```c
class Quiz
{
private:
    double score;
public:
    Quiz( );
    void Fn( );
};
```

Choose one from the following. Which member functions can carry out an assignment score = 1.0; ?

(a) Both constructor and Fn() can carry out the statement.
(b) Constructor can carry out the statement, but not Fn().
(c) Fn() can carry out the statement, but not constructor.
(d) Neither constructor nor Fn() can carry out the statement.
10. [10 points] Following is a class CounterType which represents a counter (to keep track of the number of occurrences, for example). Write the definitions of the member functions (1), (2) and (3) indicated below. Use the correct/complete syntax such that the definitions are written OUTSIDE/below the class interface part.

```cpp
class CounterType {
private:
    int count; // sets the count to 0
public:
    CounterType(); // (1) sets the count to 'init'; you may assume 'init' is non-negative.
    int get(); // returns the current count value
    void set(int cnt); // sets the counter to a count given as an argument
    void increment(); // increases the count by 1
    void increment(int inc); // (2) increases the count by 'inc'; you may assume 'inc' is non-negative.
    void decrement(); // decreases the count by 1
    void decrement(int dec); // (3) decreases the count by 'dec' only if the resulting value (after subtracting all of 'dec') does NOT yield a negative number; if it does, prints an error message and doesn't decrement; you may assume 'dec' is non-negative.
    void print(); // outputs the count to cout
}

CounterType::CounterType() {
    count = 0;
}

// Assume other member functions (besides yours) are written..

// YOUR CODE GOES HERE..
```

11. [5 points] Using the CounterType class you wrote above, write an application (main()) which does the following. Note that you may assume that the class CounterType is completely and correctly written. Also note that each step below may be accomplished by more than one line of code.

- Declare/create a CounterType object c1 by using the default constructor.
- Increment c1 by 3 by calling the method increment() three times.
- Ask the user for a number to increment c1. Then increment c1 by calling the other increment method ONCE, with an appropriate parameter.
- Declare/create another CounterType object c2 with an initial value which is the same as the current counter value of c1. Call the constructor with one int parameter to do this.
- Decrement c2 by 2 by calling the decrement method with an int parameter. Repeat this procedure 5 times, by using a loop.
- Print the current counter value of c2 to the terminal.