1. **coin changing**

a.We choose the coin with the largest denomination we can choose in each step.After one step ,we update the rest cents we should change and the numbers of coins.

Pseudo code:

```
The rest cents = n;
While(The rest cents != 0){
    Change coin=the largest denomination we can use;
    The rest cents -=Change coin;
    Coinnums++;
}
Return Coinnums;
```

prove my algorithm yields an optimal solution:

Assuming that my algorithm doesn't yield an optimal solution ,and we can find an optimal solution better than my algorithm.

We can find that there are no difference in coin order when a coin set was determined,so we can order the sequence from big to small. Let's suppose that the optimal solution is different from my algorithm's coin choice in the kth step.

We consider the kth step:Assuming we choose the denomination is A, the optimal solution choice is B. (B$\neq$A)Because we choose the largest denomination, so B is smaller than A. If so, in the optimal solution,we should choose more coins considering we can use B or smaller than B to replace A.This contradicts our assumption,and the k is arbitrary,so we can get the conclusion that my algorithm can yield an optimal solution.

b.Show that the greedy algorithm always yields an optimal solution.

The possible solution can be written as the sum of "a" as the following graph.

$$a_0c^0 + a_1c^1 + a_2c^2 + ... + a_kc^k$$

So it can be proved the same way as a. , we can assume there are some differences between the greedy algorithm and the optimal solution.We can find the greedy algorithm can have a best way in each step.If the optimal solution differs from the greedy algorithm, we should at least use c coins to replace the coin greedy algorithm used.So we can prove that the greedy algorithm always yields an optimal solution.

c. Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution.

We can set 1,7,10 cents as the e a set of coin denominations s for which the greedy algorithm does not yield an optimal solution.

Counterexample : If we want to making change for 15 cents. Using the greedy algorithm,we should use 10,1,1,1,1,1 six coins in total. But actually,the optimal solution is obvious that we can use 3 coins(7,7,1).we can find in a such set of coin denominations, the greedy algorithm does not yield an optimal solution.

2. Genetic Algorithm

The code is in Genetic Algorithm.cpp,the result is in cout result,and the Specific instructions are in the README file.

3. Answer the question 15-5 (a) in book Introduction to Algorithm (3rd Edition), chapter 15. Write down the dynamic progamming pseudo-code.

We consider the problem as the Dynamic Programing problem.We first can assume the length two words are the same.
We compare the Xi and Yj,if(i==Y.length),we can simply copy the Y's rest letters;
similarly if(j==X.length),we can simply copy the X's rest letters.
Then we compare Xi and Yj, if(Xi == Yj),we can just copy;
We consider the functions:

If we use copy,the next step we will compare Xi+1 and Yj+1; (cost1)
If we use replace,the next step we will compare Xi+1 and Yj+1; (cost2)
If we use delete,the next step we will compare Xi+1 and Yj+1; (cost3)
If we use insert ,the next step we will compare Xi and Yj+1; (cost4)
If we use twiddle,the next step we will compare Xi+2 and Yj+2;(x[i] = y[j + 1] && x[i + 1] == y[j]) (cost5)

In the end,we calculate the minimum cost between the cost1 to cost5.
the dynamic progamming pseudo-code as follows:

Pseudo code:

```
editleastdistance(x, y, i, j) {
    m = x.length;
     n = y.length;
    if( i == m) return (n - j)cost(insert);
    if (j == n) return min{(m - i)cost(delete), cost(kill)}
    if( x[i] == y[j] )
        cost1 = cost(copy) + EDIT(x, y, i + 1, j + 1);
    cost2 = cost(replace) + EDIT(x, y, i + 1, j + 1);
    cost3 = cost(delete) + EDIT(x, y, i + 1, j);
    cost4 = cost(insert) + EDIT(x, y, i, j + 1);
    if (i < m - 1 && j < n - 1)
        if (x[i] == y[j + 1] && x[i + 1] == y[j])
            o5 = cost(twiddle) + EDIT(x, y, i + 2, j + 2);
    return min{cost1,cost2,cost3,cost4,cost5};
```