1. (2 pt.)

Recursive version of strCmp

```
//Recursive version of Str_Cmp

int StrLength(char A[], int first)
{
      if (A[first]!= '\0')
            return 0;

      else
            return 1+ strLength(A, first+1);
}
```

2. (3 pt.)

Recursive version

```
bool isPalindrome(char A[],int first, int last)
{
     // Base Case #1
     if ( A[first] != A[last] )
         return false;
     // Base Cases #2, #3
     else if ( first >= last )
          return true;
     // At this point we know the characters matched
     else
          return isPalindrome(A, first+1, last-1);
}
```

3. (3 pt.)

```
#include <iostream>
using namespace std;
class ABC{
private:
      int a, b, c;
public:
      ABC(int =0, int =0, int =0);
      void print();
     // function prototype for overloaded operator `+`
      ABC operator +(ABC);
      // function prototype for overloaded operator 'cout <<'
      friend ostream& operator <<(ostream& stream, ABC );
};
ABC::ABC(int x, int y, int z)
{
      a = x; b = y; c = z;
}
// Here define  overloating operator – as a member function
ABC ABC::operator +(ABC ob)
{
     ABC tmp;
     tmp.a=a + ob.a;
     tmp.b=b + ob.b;
     tmp.c=c + ob.c;
     return tmp;
}
// overloaded cout << operator as friend function
ostream& operator <<(ostream& stream, ABC x)
{
      stream << "a = "<< x.a <<", b = "<< x.b <<", c = "<< x.c <<endl;
      return(stream);
}
int main()
{
      ABC A(20,30,40);
      ABC B(10,10,10);
      ABC C = A + B;
       cout << C;
}
```

4. (2 pt.)

```
template <class T>
T findMaxElement(T arr[], int size)
{

    T maxVal = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > maxVal) {
            maxVal = arr[i];
        }
    }
    return maxVal;
}
```