1. (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;
}
```

2. (1 pt.)
    a. Last In First Out
    b. Push(), Pop(), emptyStack(), fullStack(), Top?
    c. First In First Out
    d. enQ() deQ(), emptyQ(), fullQ(), frontQ()

3. (2 pt.)

    Recursive version of strCmp

```
//Recursive version of Str_Cmp

bool strCmp(const char *A, char *B)
{
     if (*A!=*B)
          return false;
     else if (*A=='\0'&& *B=='\0')
          return true;
     else
          return strCmp(A+1, B+1);
}
```

4. (3 pt.)

```
//a) function prototype for overloaded operator `+`
      ABC operator +(ABC);

//b) function prototype for frend overloaded operator '<<'
      friend ostream& operator <<(ostream& stream, ABC );

//c) 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;
}

//d) 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);
}
```

5. (2 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);
}
```