HW0 -Data Structure

Name: Shahad Abdullah Almuhizi
ID: 436201525
Section: 44126

Problem 1:

Wrapper class:

```
1 public class IntWrapper{
2 private int r;
3 public IntWrapper(int r){
4 this.r=r;}
5
6 public int get(){
7 return r;}
8
9 public void set(int r){
10 this.r=r;}}
```

Main:

```
1  public class Test1{
2  public static void main(String[] args){
3  int s=-2;
4  IntWrapper w=new IntWrapper(s);
5  checkModify(w);
6  s=w.get();}
7
8  public static boolean checkModify(IntWrapper w){
9  int f=w.get();
10 if (f>=0)
11 return false;
12 else
13 w.set(Math.abs(f));
14 return true;}}
```

Problem 2:

```
1 public static double evalPol(double A[], int n, double x){
2 double s=0;
3 for (int i=n; i>=0; i--)
4 s+=A[i]*(Math.pow(x,i));
5 return s;}
```

Problem 3:

1:

Generics state that the same method can be used multiple times depending on the data type of it.

The advantages of using generics are many, one of them is that it prevents runtime errors so that it shows compile errors which are easier to detect, and saves a lot of time. It does not require type casting. It offers Type safety, so only single type of object is allowed.

2:

```
1 public static <E> void exchange(E[] elements, int x, int y){
2 E temp=elements[x];
3 elements[x]=elements[y];
4 elements[y]=temp;}
```

3:

```
1 public static <E> void count(E[] elements, E c){
2 int count=0;
3 for (int i=0;i<elements.length;i++)
4 if (elements[i]==c)
5 count++;
6 return count;}
```

4:

```
1 public static <T> void reverse(T[] A, int n){
2 for (int i=0; i<n-1; i++){
3 T temp=A[n-1];
4 A[n-1]=A[i];
5 A[i]=temp;
6 n--;}}
```

Problem 4:

```
1 public class GArray <T> {
2 public GArray(int size) {
3 // size is the length of the array
4 T[] arr =(T[]) new Object[size];}
5
6
7 public T get(int i) {
8 // Return the element at position i
9 if (i<arr.length&&i>=0)
10 return arr[i];
11 return null;}
12
13
14 public void set(int i, T e) {
15 // Put e at index i
16 if (i<arr.length&&i>=0)
17 arr[i] =e;}}
```

Problem 5:

```
1 public static <T, U> GArray<Pair<T,U>> pair(T[] A, U[] B, int n){
2
3 GArray<Pair<T,U>> arr= new GArray <Pair<T,U>>(n);
4 if ((n>A.length)&&(n>B.length))
5 return null;
6
7 for (int i=0; i<n; i++)
8 arr.set(i, new Pair<T,U>(A[i],B[i]));
9 return arr;}
```

Problem 6 :

1:

```
1 public static <T extends Comparable<T>> void sort(T[] A, int n, boolean incr) {
2 for( int i=0;i<n ;i++)
3
4 if (incr==ture){
5 for (int j=0; j<n; j++)
6 if (A[i].compareTo(A[j])<=0){
7 T temp=A[j];
8 A[j]=A[i];
9 A[i]=temp;}}
10 else{
11 for (int j=0; j<n; j++)
12 if (A[i].compareTo(A[j])>=0){
13 T temp=A[j];
14 A[j]=A[i];
15 A[i]=temp;}}}
```

2:

```
1 public static void test(){
2 String arr[]={"A","B","C"};
3 sort(arr,2,true);}
```