Test zum Stoff MPGI 2- Bearbeitungszeit 75 Minunten

Bemerkung:

Überall dort wo Startknoten benötigt werden soll der Knoten A als Startknoten dienen.

1 Aufgabe *(10 Punkte)*

Aussage	wahr	falsch
Das Hoare Kalkül beweist die totale Korrektheit eines Algorithmus‘.		
Die topologische Sortierung ist immer eindeutig.		
Beim Sondieren darf die Anzahl der Schlüssel die Größe der Hashtabelle nicht überschreiten.		
Bei der Tiefensuche werden zunächst alle (noch nicht besuchten) Nachbarn besucht und dann die Nachbarn der Nachbarn		
Arrays erlauben aufgrund der Indizierung einen sequentiellen Zugriff auf die Elemente		
Neben dem Dijkstra Algorithmus bestimmt auch der Bellman-Ford Algorithmus die kürzesten Wege in einem Graphen.		
Der Prim Algorithmus bestimmt den minimalen Spannbaum, in dem er die Kanten mit den kleinsten Gewichten zuerst betrachtet.		
Die Gesamtkomplexität des Kruskalalgorithmus beträgt $O(\|E\|\log\|E\|)$.		
Der Ford-Fulkerson-Algorithmus findet einen maximalen Fluss in einem Flussgraphen		
Der Wert des maximalen Flusses in einem Flussgraphen ist gleich der Kapazität des minimalen Schnittes.		

2 Aufgabe - Listen *(5 Punkte)*

1. Gegeben ist die Definition einer einfach verketteten Liste.
   Schreibt diese in eine zyklisch verkettete Liste um. *(2 Punkte)*

Liste.java

```
public class Liste<T>
{
        private class ListElem<T>
        {
                public T value;
                public ListElem<T> next=null;

                public ListElem(T v)
                {
                        value=v;
                }
        }

        public ListElem<T> head;

        public Liste(ListElem<T> el)
        {
                head=el;
        }

        /*
                returns true if elem is the last element of the List
                false otherwise
        */
        public boolean isLast(ListElem<T> elem)
        {
                return (elem.next==null);
        }
}
```

2. Schreibt eine Methode `public void remove(int i)`, die das i-te Element aus einer doppeltverketteten Liste löscht. *(2 Punkte)*

3. Gegeben ist eine Methode. Gebt an was sie macht und welchen Aufwand sie hat. *(2 Punkte)*

```
public void wasMacheIch()
{
        ListElem<T> curr= this.head();

        if(head==null) System.out.println("Liste ist leer.");

        while(curr!=null)
        {
                System.out.print(curr.value() + "-");
                curr=curr.next();
        }
        System.out.println("\n");
}
```

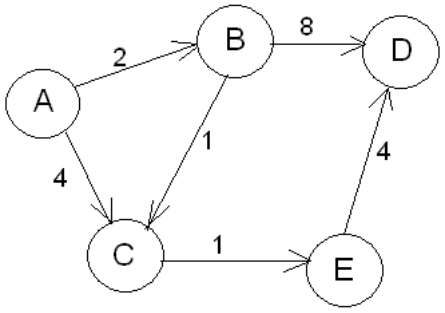
3 Aufgabe - Arrays *(3 Punkte)*

Schreibt eine Funktion *anzElements* die ein Array mit Listen (mit Integerwerten) übergeben bekommt und ein Array mit den Längen der einzelnen Listen zurückgibt.

4 Aufgabe - Graphen *(6 Punkte)*

1. Gegeben ist eine Menge Knoten V und eine Menge Kanten E. Erstellt daraus einen Graphen.*(1 Punkt)*
   V={A,B,C,D} ; E={(A, B, 1), (A, C, 7),(B, A, 2),(B, D, 3),(D, A, 5)}

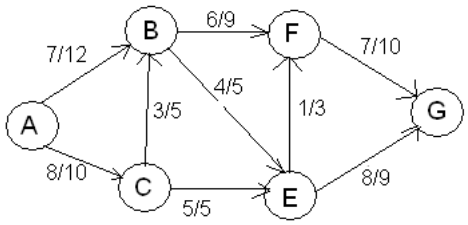
2. Gegeben ist der folgende Graph. Findet mit Hilfe des Dijkstra Algorithmus den kürzesten Weg vom Startknoten A zu allen anderen Knoten.*(Punkte 3)*



3. Ist der in 3.2 gegebene Graph topologisch sortierbar?
   Wenn ja, gebt die Sortierung an. Wenn nein, begründet warum nicht. *(Punkte 2)*

5 Aufgabe - Flüsse *(8 Punkte)*

Gegeben ist der folgende Graph.



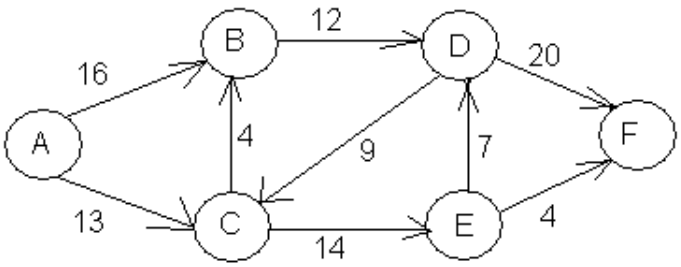
1. Handelt es sich hierbei um einen gültigen Flussgraph?
   Wenn nein, begründet warum nicht. Wenn ja, gebt den Fluss von A nach G an. *(1 Punkte)*

2. -

3. Gebt den dazugehörigen Restflussgraph an. *(1 Punkt)*

4. Gebt in dem Restflussgraphen einen Flussvergrößerungspfad (von A nach G) und dessen Restkapazität an. *(2 Punkte)*

Gegeben ist der folgende Restflussgraph.



1. Simuliert den Ford-Fulkerson Algorithmus anhand des obigen Graphen *(3 Punkte)*
2. Zeichnet den minimalen Schnitt ein. *(1 Punkt)*

6 Aufgabe - Hashing *(2 Punkte)*

Fügt die Schlüssel [4,7,1] in die noch leere Hashtabelle ein.

Gegeben ist dazu die Hashfunktion h(x)= x mod 3 und die Sondierungsfunktion g(x,j) = (x*j+1) mod 3.

7 Aufgabe - Sortieralgorithmen *(6 Punkte)*

Gegeben sind die Sortieralgorithmen A und B.

Sortierung.java

```
public void A()
{
        for (int s=0; s<array.length-1; s++)
        {
                int index= helpA(s);
                swap(array,s,index);
                for(int i=0;i<array.length; i++)
                {
                        System.out.print(" - " + array[i]);
                }
                System.out.print("\n");
        }
}

protected int helpA(int s)
{
        int m=array[s];
        int in=s;

        for(int i=s+1; i < array.length; i++)
        {
                if(array[i]<=m)
                {
                        m=array[i];
                        in=i;
                }
        }
        return in;
}

public void B(int[] a,int start, int end)
{
        for(int i=start; i<=end;i++)
        {
                for(int j=end-1; j>=i; j--)
                {
                        if(a[j]>a[j+1]) swap(a,j,j+1);
                }
        }
}
```

1. Gebt jeweils an um welchen Algorithmus es sich handelt und welche Komplexität
   der jeweilige Algorithmus hat. *(4 Punkte)*
2. Gebt jeweils an, ob sie in situ oder ex situ und stabil sind. *(2 Punkte)*

8 Aufgabe - Suchbäume *(10 Punkte)*

Gegeben ist die folgende Liste 8→ 3→ 1→ 5→4→ 6→ 9→null

1. Fügt die Listenelemente schrittweise nacheinander in einen noch leeren binären Suchbaum ein. *(2 Punkte)*
2. Ist der im letzten Schritt entstandene Baum balanciert? *(1 Punkt)*
3. Ist er linksvoll oder rechtsvoll oder keins von beidem? *(1 Punkt)*
4. Gebt die Postorder und Preorder Sortierung an. *(1 Punkt)*
5. In welcher Reihenfolge müssten sich die Elemente in der obigen Liste befinden, damit man nach dem Einfügen aller Elemente einen vollständigen Baum erhält? *(1 Punkt)*
6. Wie kann man einen mit Hilfe eines Suchbaumes ein Array sortieren? Erklärt euer Vorgehen.*(2 Punkte)*
7. Schreibt eine Funktion die auf jeden Wert in einem Baum mit Integerwerten 5 draufaddiert. *(2 Punkt)*
8. Gegeben sind die folgenden Sortierungen:
   Preorder: 7, 3, 1, 4, 5, 6, 13, 12, 10
   Inorder: 1, 3, 4, 5, 6, 7, 10, 12, 13
   Rekonstruiert daraus den Suchbaum.

9 Aufgabe - Korrektheit *(7 Punkte)*

9.1 Hoare Kalkül *(2 Punkte)*

Wie lauten die Verzweigungs- und Zuweisungsregel?

9.2 Korrektheitsbeweis mit dem Hoare Kalkül *(5 Punkte)*

Gegeben ist der folgende Methodenausschnitt.

Beweist mittels Hoare Kalkül die Korrektheit dieses Ausschnittes.

```
public   int foo (int a,   int b)
{
         if (a < b)
                            return b-a;
         else
                            return a-b;
}
```