Heimadæmi 1 – Tölvunarfræði 2

1. [Kennslufræði] Hér er grein úr Scientific American sem lýsir fimm námsaðferðum sem virka vel og nokkrum öðrum sem virka ekki eins vel. Notar þú einhverja af "góðu" aðferðunum? Hver er þín reynsla af aðferðunum í greininni? Skrifið nokkrar setningar (ekki ritgerð!).

SVAR:
Hef gjarnan notað quizlet eða Anki, þar er farið ó flashcards sem virkar vel fyrir ákveðnar námsgreinar finnst mér, hentar vel ef á að muna skilgreiningar, t.d. tungumál, anatomy og fleira.

2. [Kennslufræði] Hér er stutt grein úr tímaritinu Science um áhrif upprifjunarnáms á námsárangur undir streituálagi. Lýsið niðurstöðunum í Mynd 1 í greininni í nokkrum orðum.

SVAR:
Gerð 2 test, test 1 beint eftir að fólk var fyrir streituálagi, test 2 var 25 mín eftir streituálag. Einnig var skipt í SP = Study practise og RP = Retrieval practice. Fyrir test 1 var betri eða sambærileg memory performance hjá hópum SP og RP. Fyrir test 2 hafði stress áhrif neikvæð áhrif á SP en jákvæð á RP. Þannig má segja að Retrieval practise er betri leið til að læra undir álagi í lok dags.

3. [Java upprifjun] Skrifið Java forritið TLeikur.java fyrir eftirfarandi teningaleik: kastað er 10-hliða tening og á meðan gildin sem koma upp fara strangt hækkandi þá eru þau lögð saman. Forritið hættir keyrslu um leið og nýjasta gildið er lægra eða jafnt og síðasta gildi. Til dæmis, ef fyrsta kast gefur 4, annað kast 7 og þriðja kast 2, þá er prentað út 11 (því 4+7 = 11). Athugið að fyrsta gildið er alltaf með í summunni. Vísbending: Aðferðin StdRandom.uniformInt(N) býr til slembiheiltölu á bilinu [0, N−1].

SVAR:

```java
import edu.princeton.cs.algs4.*;

public class TLeikur {
    public static void main(String[] args) {
        int summa = 0;
        int sidastaGildi = 0;
        int nyttGildi;

        while (true) {
            nyttGildi = StdRandom.uniformInt(1, 11);
            if (nyttGildi <= sidastaGildi) {
                break;
            }
            summa += nyttGildi;
            sidastaGildi = nyttGildi;
        }

        System.out.println(summa);
    }
}
```

4. [Hlaðar] (sbr. dæmi 1.3.3 í bók) Notendaforrit framkvæmir blöndu af push og popaðgerðum. push-aðgerðin setur heiltölurnar 0, 1, ..., 9 í þessari röð á hlaðann, en popaðgerðirnar prenta út skilagildið. Sýnið hvenær pop-aðgerðirnar koma í röð aðgerða í eftirfarandi úttaksrunum. Fyrir þær úttaksrunur sem ekki eru mögulegar, útskýrið hvers vegna:

   a. 4 3 5 2 1 6 7 0 9 8
   b. 2 1 5 6 4 3 9 8 7 0
   c. 0 2 4 3 6 7 1 8 9 5

SVAR: notum – fyrir pop.

a. 0,1,2,3,4,-,-,5,-,-,-,6,7,-,-,8,9,-,-
b. 0,1,2,-,-,3,4,5,-,6,-,-,-,7,8,9,-,-,-,-
c. 0,-,1,2,-,3,4,-,-,5,6,-,7,-, hér er ekki hægt að fá 1 því næsti pop er 5.

5. [Tengdir listar] Notið tengda listann fyrir strengi: LinkedListOfStrings.java og bætið við hann aðferðinni **count(S)** sem fer í gegnum tengda listann og skilar fjölda tilvika af strengnum **S** sem koma fyrir í listanum. Ef **S** kemur ekki fyrir í listanum þá skilar aðferðin 0. Prófunarfall klasans býr til tengdan lista úr strengjunum sem eru á skipanalínunni. Breytið prófunarfallinu þannig að það að það lesi auk þess einn streng af staðalinntaki og prenti út fjölda tilvika af honum í tengda listanum. Sýnið aðferðina og skjámynd af keyrslu. Dæmi um keyrslu (innsláttur notanda er rauður):

   % java LinkedListOfStrings A B A D C
   C->D->A->B->A->
   Search string:
   A Number of instances: 2

SVAR:

```java
import java.util.NoSuchElementException;
import edu.princeton.cs.algs4.*;

public class LinkedListOfStrings {
    private int N; // size of list
    private Node first; // first node of list

    // helper Node class
    private static class Node {
        private String item;
        private Node next;
    }

    public LinkedListOfStrings() {
        N = 0;
        first = null;
    }
```

```java
// is the list empty?
public boolean isEmpty() {
    return first == null;
}

// number of elements on the stack
public int size() {
    return N;
}

// add an element to the front of the list
public void addFront(String item) {
    Node oldfirst = first;
    first = new Node();
    first.item = item;
    first.next = oldfirst;
    N++;
}

// delete and return the first item in the list
public String delFront() {
    if (isEmpty())
        throw new NoSuchElementException("No items in list");
    String item = first.item; // save item to return
    first = first.next; // delete first node
    N--;
    return item; // return the saved item
}

// print out the list
public void printList() {
    Node x = first;
    for (int i = 0; i < N; i++) {
        StdOut.print(x.item + "->");
        x = x.next;
    }
    StdOut.println();
}

// Counts instances of String S in the list
public int count(String S) {
    Node current = first;
    int count = 0;
    while (current != null) {
        if (current.item.equals(S)) {
            count++;
        }
        current = current.next;
    }
    return count;
```

```
    }

    // test client
    public static void main(String[] args) {
        LinkedListOfStrings list = new LinkedListOfStrings();
        // Adding elements to the list from command-line arguments
        for (int i = args.length - 1; i >= 0; i--) {
            list.addFront(args[i]);
        }

        // Printing the list
        StdOut.println("Number of items: " + list.size());
        list.printList();

        // Reading a string from standard input
        Scanner scanner = new Scanner(System.in);
        StdOut.print("Search string: ");
        String searchStr = scanner.nextLine();

        // Printing the number of instances of the string
        StdOut.println("Number of instances: " + list.count(searchStr));
        scanner.close();
    }
}
```