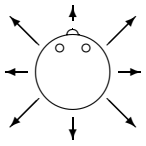
| Západočeská univerzita v Plzni | | Katedra informatiky a výpočetní techniky | |
|---|---|---|---|
| Písemná zkouška z předmětu: | | Programovací techniky – KIV/PT | |
| Zkoušející: | | Ing. Pavel Mautner, PhD. | |
| Akademický rok: | 2005/06 | Datum: | |
| Příjmení: | | Křestní jméno: | |
| Osobní číslo: | | Obor: | |
| Celkový počet dosažených bodů (max. 100): | | | |

| Úloha č.: | 1 | 2 | 3 | 4 | 5 | 6 | součet | cvičení | celkem | známka |
|---|---|---|---|---|---|---|---|---|---|---|
| max. bodů: | 20 | 12 | 20 | 20 | 14 | 14 | 100 | | | |
| skut. bodů: | | | | | | | | | | |

**UPOZORNĚNÍ:**

a) Písemná zkouška obsahuje 6 úloh, jejichž řešení musí být vepsáno do připraveného formuláře. Pokud někomu nebude vymezený prostor dostačovat, lze využít též druhé (čisté) strany formulářů s tím, že bude zřetelně vyznačeno, ke které úloze řešení přísluší.

b) Do obrázku napravo doplňte jména sousedů sedících okolo Vás v naznačených směrech:



## 1. Implementace datových typů [ 20 bodů ]]

Mějme v jazyce Java definovanou třídu **Node**, která implementuje obecný prvek jednosměrně zřetězeného seznamu.

```
public class Node {
  private Node next;                                  // ukazatel na nasledujici prvel
  private Object element;                             // ukazatel na datovy objekt

  public Node() { next=null; element=null; }          // vytvori prazdny prvek
  public Node(Object elm) { next=null; element=elm; }
  public void set_Next(Node snext)  { next=snext;  }  // nastavi ukazatel next na snext
  public void set_Elm(Object elm)    { element=elm; } // nastavi element na elm
  public Node get_Next() { return next; }             // vraci ukazatel next;
  public Object get_Elm() {return element; }          // vraci ukazatel na dat. objekt }
```

Dále mějme definovanou třídu **Stack** implementující zásobník jako seznam jednosměrně zřetězených prvků typu **integer**.

Do níže uvedené kostry třídy **Stack** doplňte:

1. konstruktor **Stack**, který vytvoří prázdný zásobník [ 5 bodů ]
2. metodu **push**, která vloží prvek na zásobník [ 5 bodů ]
3. metodu **pop**, která vybere prvek ze zásobníku a vrátí odkaz na tento prvek [ 5 bodů ]

Co se vypíše na obrazovku po spuštění metody **main**? ..... [ 5 bodů ]

```
public class StackL {
    private Node top;  // ukazatel na vrchol zasobniku
    private int sz;    // pocet objektu v zasobniku

    // konstruktor tridy - vytvori prazdny zasobnik
    //  !!!!!!!!!!!!! DOPLNIT !!!!!!!!!!!!!!!!!!!!!

    public StackL() {




    }

  // Metoda vlozi prvek na zasobnik
  //  !!!!!!!!!!!!! DOPLNIT !!!!!!!!!!!!!!!!!!!!!

  public void push(Object o){









    }

  //Metoda vybere prvek ze zasobniku a vrati ukazatel na tento prvek
  // je-li zasobnik prazdny vyvola vyjimku StackEmptyException
  //   !!!!!!!!!!!!! DOPLNIT !!!!!!!!!!!!!!!!!!!!!

   public Node pop()throws StackEmptyException {









        }



    //Metoda testuje zda je zásobník prázdný a pokud ano vrací true
    // opačném případě vrací false
    public boolean isEmpty() { return(top==null); }

    public int size(){ return(sz);} // vraci pocet objektu v zasobniku

    //Metoda vraci hodnotu prvku z vrcholu zasobniku, prvek zustava na zasobniku
    public Object top() throws StackEmptyException { ..............  }


    public class StackEmptyException extends RuntimeException {
        public StackEmptyException(String err) {
            super(err);
        }
    }

    public static void main(String[] args)  {
        StackL st=new StackL();
        int [] a={10, 12, 15, 21, 33};
        for (int i=0; i<a.length; i++)
        st.push(new Integer(a[i]));
        int n=st.size();
        System.out.print(n+"  ");
        while (!st.isEmpty())
        System.out.print(st.pop().getElm().toString()+"  ");
    }
}
```

**2. Řídicí struktury programovacích jazyků** [ 12 bodů ]

Na protilehlou volnou zadní stranu předchozího listu zapište v programovacím jazyce Java funkční metodu (public static) `Fakt(n)`, která

a) **bez rekurze** (iteračně) [ 5 bodů ]

b) **rekurzivně** [ 5 bodů ]

vyčíslí faktoriál čísla $n \in \langle 0, 20 \rangle$ .

c) Do níže uvedené tabulky slovně i $\mathcal{O}$-notací vyjádřete, jaká je algoritmická i paměťová složitost výše uvedeného výpočtu faktoriálu: [ 2 body ]

| | algoritmická - slovně | algoritmická - $\mathcal{O}$-notací | paměťová - slovně | paměťová - $\mathcal{O}$- notací |
|---|---|---|---|---|
| ad a) | | | | |
| ad b) | | | | |

**3. Vyhledávání dat** [ 20 bodů ]

**A)** Určete, kolik operací porovnávání znaků musí provést

a) jednoduchý algoritmus přímého vyhledávání (brute-force) [ 2 body ]

b) Boyer - Mooreův algoritmus [ 5 bodů ]

c) Knuth - Moris - Prattův algoritmus [ 5 bodů ]

pro nalezení slova `MALEBNOU` ve znakovém řetězci (prohledávaném textu):

`MALIR_MALOVAL_MALEBNOU_KRAJINU`

**i)**..... porovnání **ii)**..... porovnání **iii)**..... porovnání

**B)** Do předtištěného pole doplňte pro uvedené znaky hodnotu funkce **Last(x)** použité u Boayer - Moorova algoritmu [ 4 body ]

| x | _ | A | B | E | I | J | K | L | M | N | O | R | U | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Last(x) | | | | | | | | | | | | | | |

**C)** Do předtištěného pole doplňte hodnoty chybové funkce **F(k)** KMP algoritmu pro hledané slovo `MALEBNOU` [ 4 body ]

| k | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| F(k) | | | | | | | | |

**4. Stromové struktury** [ 20 bodů ]

Mějte dánu následující posloupnost datových položek identifikovatelných celočíselnými klíči:

**92, 16, 56, 74, 12, 49, 89, 68, 82, 58, 33, 99, 62, 80, 25, 39, 21**

Posloupnost položek reprezentujte zadanými typy stromů (pro jednoduchost znázornění reprezentujte položky jen uvedenými hodnotami klíčů, tam, kde je to nutné, dodržte výše zadané pořadí položek). Po vytvoření stromu odeberte v uvedeném pořadí prvky s klíči **92, 74, 12**. (Při rušení prvku ve vnitřním uzlu BVS a AVL stromu nahrazujte rušený prvek jeho symetrickým předchůdcem).

**a)** Reprezentace **BVS** stromem: ( 5 bodů )

**b)** Reprezentace **AVL** stromem (prvky vkládejte postupně a pokud je to nutné provádějte vyvážení stromu po každé operaci vložení): ( 5 bodů )

**c)** Reprezentace **B** stromem řádu m=3 ( uzel obsahuje 3 ukazatele) ( 5 bodů )

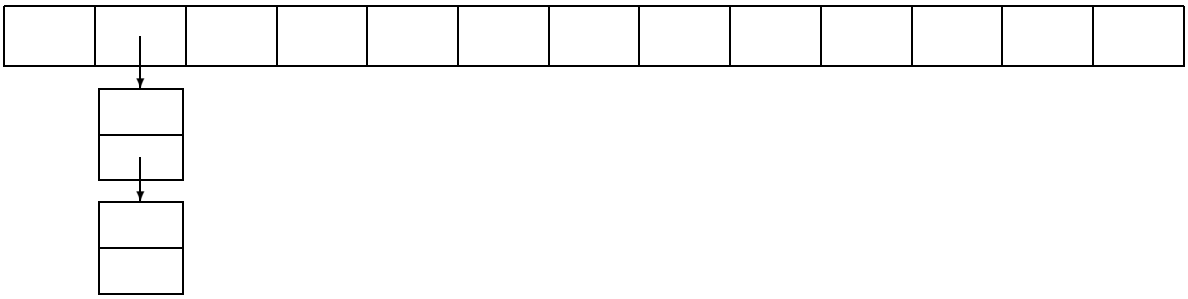
**d)** Reprezentace **B** stromem řádu m=5 (uzel obsahuje 5 ukazatelů): ( 5 bodů )

**5. Tabulky s rozptýlenými položkami** [ 14 bodů ]

**a)** Do níže vyobrazené struktury, která implementuje hash-tabulku s vnějším zřetězením s ukládáním synonymických položek do seznamů zřetězených prvků (metoda "scattered index"), zařaďte (zakreslete naznačeným způsobem) položky s níže uvedenými celočíselnými klíči v pořadí: ( 6 bodů )

134 196 275 313 415 422 460 563 630 738 783 857

Jako rozptylovou funkci pro prvotní přístup do jednotlivých seznamů použijte zbytek po celočíselném dělení hodnoty klíče příslušným základem.



**b)** vyčíslete střední algoritmickou složitost vyhledávání položek (v počtu operací porovnání klíčů potřebných pro vyhledání položky) v takto vytvořené implemetaci tabulky:

střední počet operací porovnání klíčů: . . . . . ( 3 body )

**c)** Lze v tabulkách s rozptýlenými položkami vyhledávat položky metodou binárního vyhledávání ? Vhodnou odpověď zakroužkujte, nehodící se zřetelně škrtněte ! ( 2 body )

**ano** **ne**

Svoji odpověď patřičně zdůvodněte:

**6. Komprese dat** [ 14 bodů ]

**a)** Pro znakový řetězec

HRADEC _ KRALOVE

kde znak _ značí mezeru, vytvořte Huffmanův kódovací strom; aby Vámi vytvořený strom byl jednoznačný, důsledně dodržujte řazení znaků vkládaných do stromu podle četností jejich výskytu v řetězci a uvnitř tříd znaků se stejnou četností pak dodržujte abecední řazení znaků s tím, že _ (mezeru) řaďte **na začátek abecedy**, jak je dáno kódovou tabulkou ASCII kódu: ( 10 bodů )

**b)** Vámi vytvořeným Huffmanovým kódovacím stromem zakódujte slovo DRAHA : ( 2 body )

Zakódovaná posloupnost: . . . . . .

**c)** Tímtéž stromem dekódujte komprimovaný řetězec (bankovní operace) ( 2 body )

1011100010011100110

Originální posloupnost: . . . . . .