Informatik I D-ITET Master 20.8.2014 *F. Friedrich*

Name, Vorname: .....

Legi-Nummer: .....

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ausführen konnte und dass ich die allgemeinen Richtlinien gelesen und verstanden habe.

I confirm with my signature that I was able to take this exam under regular conditions and that I have read and understood the general guidelines.

Unterschrift:

Allgemeine Richtlinien:

a) Dauer der Prüfung: 60 Minuten.

b) Erlaubte Unterlagen: Lehrbuch C++, Zusammenfassung beliebiger Länge, Ausdruck der Präsentationen der Vorlesung. Keine elektronischen Geräte!

c) Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.

d) Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben. Ungültige Lösungen deutlich durchstreichen. Korrekturen bei Multiple-Choice Aufgaben unmissverständlich anbringen.

e) Falls Sie sich durch irgendjemanden oder irgendetwas gestört fühlen, melden Sie dies sofort der Aufsichtsperson.

f) Wir sammeln die Prüfung zum Schluss ein. Wichtig: stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einem Assistenten eingezogen wird. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: bitte melden Sie sich lautlos und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 15 Minuten vor Prüfungsende möglich.

g) Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen. Es darf zu einer Zeit immer nur eine Studentin oder ein Student zur Toilette.

h) Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

General guidelines:

Exam duration: 60 minutes.

Permitted examination aids: textbook on C++, lecture summary of arbitrary length, printouts of the lecture slides. No electronic devices!

Use a pen (black or blue), not a pencil. Please write legibly. We will only correct solutions that we can read.

All solutions must be written directly onto the exercise sheets in the provided boxes. Invalid solutions need to be crossed out clearly. Corrections to answers of multiple choice questions must be provided without any ambiguity.

If you feel disturbed by anyone or anything, immediately let the supervisor of the exam know this.

We collect the exams at the end. Important: you must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: please contact us silently and we will collect the exam. Handing in your exam preliminarily is only possible until 15 minutes before the exam ends.

If you need to go to the toilet, raise your hand and wait for a supervisor. Only one student can go to the toilet at a time.

We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.

Aufgabe	1	2	3	4	5	6	$\sum$
Punkte							
Maximum	21	16	15	16	16	16	100

1 Elementares *(21 Punkte)*

⊛ $12 = 6 \times 2$

a) Bestimmen Sie den finalen Typ (int, bool, float oder double) und Wert der folgenden C++Ausdrücke.

Determine the final type (int, bool, float or double) and value of the following C++expressions.

Voraussetzung	Ausdruck / expression	Typ / type	Wert / Value
–	`1 / 4 * 4.0f`	float	0
`int x;`	`x = 0.9 + 1.0 / 4 * 4`	int	1

`float x;`
`int y = 7;`
`int z = 5;`
	`x = ++y / --z ;`	float	2

`int k = 14;` `int i = 6;`	`k % i < k / i`	bool	false
–	`false - true`	int	-1
–	`1u - 10 > 0`	bool	true

b) Geben Sie für jedes der drei folgenden Code-Fragmente die Folge von Zahlen an, die das Fragment ausgibt

Provide the sequence of numbers that is generated by each of the following three code fragments.

(i)
```
for (int i=1; i<100; i*=2)
    std::cout << i++ << "␣";
```

Ausgabe: 1 4 10 22 46 94

⊛ 3

(ii)
```
int a=38;
for (int i=0; i<8; ++i)
    {
        std::cout << (a % 2);
        a /= 2;
    }
```

Ausgabe: 01100100

⊛ 3

(iii)
```
unsigned int x = 1;
do {
    std::cout << x << "␣";
    x = (3 * x + 2) % 7;
} while (x != 1);
```

Ausgabe: 1 5 3 4 0 2

⊛ 3

Ende Aufgabe 1

2 Datenstrukturen *(16 Punkte)*

Gegeben seien folgende Daten, eine Sequenz von Zahlen

Consider the following data, a sequence of numbers

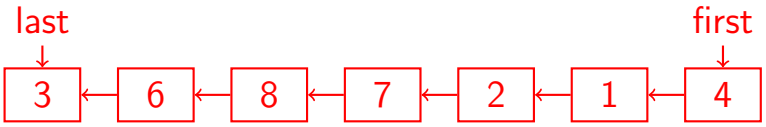
4, 1, 2 ,7, 8 ,6 ,3

Vervollständigen Sie folgende Figuren, so dass sie deutlich machen, wie die jeweilige Datenstruktur nach Einfügen der gegebenen Daten in obiger Reihenfolge (von links nach rechts) aussieht. Vergessen Sie nicht, die Daten (Zahlen) auch einzutragen.

Complete the following figures such that they illustrate the respective data structures after insertion of the given data in the order above (from left to right). Do not forget to enter the data (numbers) also.

a) Warteschlange (FIFO) mit verketteter Liste

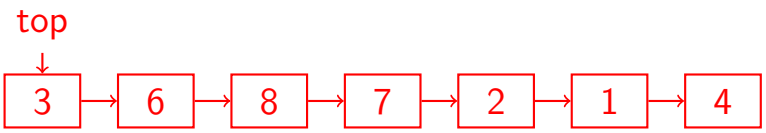
Queue (FIFO) with linked list



⊛ 2

b) Stack mit verketteter Liste

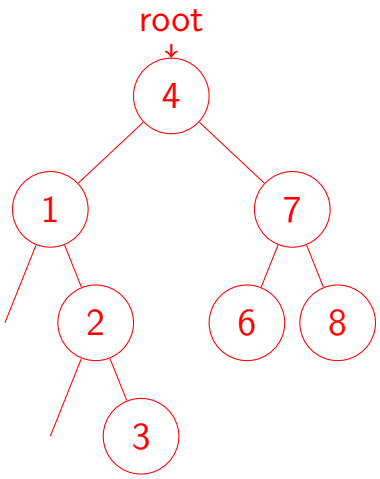
Stack with linked list



⊛ 4

c) Vervollständigen Sie folgende Zeichnung so, dass sie die Datenstruktur `Tree` des Programmes auf der folgenden Seite nach Ausführung von main wiedergibt.

Complement the following figure such that it represents the data structure `Tree` *of the program of the next page after execution of the main fucntion.*



⊛ 5

d) Vervollständigen Sie die Funktion `Sum` in folgendem Code so, dass ein Aufruf von `Tree.GetSum()` die Summe der Werte aller Knoten des Baumes zurückgibt.

Complete function `Sum` in the subsequent code such that a call to `Tree.GetSum()` returns the sum of values of all nodes of the tree.

```
#include <iostream>

struct Node{
    Node *left, *right;
    int value;

    Node (const int v): left(0), right(0), value(v) {};
};

void PutNode(const int value, Node* & node) {
    if (node == 0)
        node = new Node(value);
    else if (value < node->value)
        PutNode(value, node->left);
    else
        PutNode(value, node->right);
}

int Sum(Node* const node) {
   if (node == 0) return 0;
```

⊛ 5

```
    return (node->value) + Sum(node->left) + Sum(node->right);
}

class Tree{
    Node* root;
public:
    Tree(): root(0) {};

    void Put(const int value) { PutNode(value, root); }

    int GetSum() const { return Sum(root); }
};

int main () {
    Tree tree;
    const int arr[] = {4, 1, 2 ,7, 8 ,6 ,3};
    for (int i = 0; i<7; ++i)
        tree.Put(arr[i]);
}
```

Ende Aufgabe 2

3 Objektorientierte Programmierung *(15 Punkte)*

a) Schreiben Sie die erzeugte Ausgabe des folgenden Programmes in den Kasten.

Write the output generated by the following program into the box.

```
#include <iostream>

struct A {
    int count;

    A(){count = 0;}

    void Hello (){
        std::cout << "hello A " << count++ << "\n";
    }

    virtual void Goodbye (){
        std::cout << "bye A "  << count++ << "\n";;
    }

};

struct B:A {
    void Hello (){
        std::cout << "hello B " << count++ << "\n";
    }

    void Goodbye (){
        std::cout << "bye B " << count++ << "\n";
    }
};

int main()
{
    B b;
    A c = b;

    b.Hello();
    c.Hello();
    c.Goodbye();
    b.Goodbye();

    B* pb = new B;
    A* pc = pb;

    pb->Hello();
    pc->Hello();
    pc->Goodbye();
    pb->Goodbye();

    return 0;
}
```

⊛ 8

hello B 0
hello A 0
bye A 1
bye B 1
hello B 0
hello A 1
bye B 2
bye B 3

b) Das folgende Programm ist nicht vollständig. Ergänzen Sie es so, dass es kompiliert werden kann und dass der Spieler mit Würfel `playerTwoDice` in der Main-Funktion mit grösserer Wahrscheinlichkeit eine sechs würfelt.

The following program is not yet complete. Complement it such that it can be compiled and such that the player with the dice `playerTwoDice` *in the main function tosses a six with higher probability.*

```
#include <ctime>
#include <cstdlib>
#include <iostream>

struct Dice {
    virtual int Toss();
};

struct FairDice : Dice {
    int Toss() {
        return 1+ std::rand() % 6;
    };
};

struct LoadedDice : Dice {
    int Toss() {
        if (std::rand() % 2)
            return 6;
        else
            return 1 + std::rand() % 6;
    };
};

int main () {
    std::srand (std::time (0)); // init random generator

    Dice* const  playerOneDice = new FairDice();
    Dice* const  playerTwoDice = new LoadedDice();

    const int resultOne = playerOneDice->Toss();
    const int resultTwo = playerTwoDice->Toss();

    if (resultOne > resultTwo)
        std::cout << "player one wins";
    else if (resultTwo > resultOne)
        std::cout << "player two wins";
    else
        std::cout << "tie.";
    return 0;
}
```

⊛ 2

⊛ 2

⊛ 3

Ende Aufgabe 3

4 Sortierte Ziffern *(16 Punkte)*

Eine nicht negative ganze Zahl soll darauf hin untersucht werden, ob deren Ziffern aufsteigend (mit $\leq$) sortiert sind. Wir nennen im folgenden eine solche Zahl **Ziffer-sortiert**.
Beispiele:

For a non-negative integer number it shall be determined if the digits are increasingly (with $\leq$) sorted. In the following we refer to such a number as **digit-sorted**.
Examples:

- 1234 ist Ziffer-sortiert — *1234 is digit-sorted*
- 1222 ist Ziffer-sortiert — *1222 is digit-sorted*
- 21 ist **nicht** Ziffer-sortiert — *21 is* **not** *digit-sorted*
- 2 ist Ziffer-sortiert (es gilt ja auch $2 = 02$) — *2 is digit-sorted (since also $2 = 02$)*

a) Ergänzen Sie den folgenden Code so, dass die Funktion `IsDigitSorted` zurückgibt, ob die Zahl Ziffer-sortiert ist.

Complement the following code such that the function `IsDigitSorted` *returns true if the number is digit-sorted and false otherwise.*

```
// pre: non-negative number
bool IsDigitSorted(int number)
{
    while (number >= 10){
        int remainder = number % 10;
        number = number / 10;
        if (number % 10 > remainder) return false;
    }
    return true;
}
```

⊛ 3

b) Ergänzen Sie folgende main-Funktion so, dass Sie, obigen Code verwendend, die **Anzahl** Ziffer-sortierter Ganzzahlen von 0 bis 999 ausgibt.

Complement the following main-function such that it prints, using the code above, the **number** *of digit-sorted integers from 0 to 999.*

```
int main () {
    int number = 0;
    for (int i = 0; i<1000; ++i)
        if (IsDigitSorted(i)) ++number;
    std::cout << "number of digit-sorted values: " << number << "\n";
}
```

⊛ 3

c) Wenn Sie oben genanntes Verfahren auf Binärzahlen anwenden, welche Zahlen sind dann aufsteigend sortierte Ganzzahlen? Geben Sie einen mathematischen Ausdruck an.

If you apply the method above to binary numbers, which numbers are then digit-sorted? Provide a mathematical expression.

$2^n - 1$ für ein $n \geq 0$

⊛ 2

d) Offensichtlich ist die oben gewählte iterative Methode zur Berechnung der Anzahl Ziffer-sortierter Zahlen wenig effizient. Vervollständigen Sie folgende mathematische **Rekursionsformel** für die Berechnung der Anzahl Ziffer-sortierter Zahlen mit n Ziffern, deren führende Ziffer grösser gleich d sind.

Obviously the chosen iterative version from above for the computation of the number of digit-sorted integers is not very efficient. Complement the following mathematical **recursion formula** *for the computation of the number of digit-sorted numbers with given number of digits n and value of the leading digit being greater or equal d.*

$$S(n,d) = \begin{cases} \text{10-d} & \text{falls } n = 1 \\ \sum_{i=d}^{9} \text{S(n-1,i)} & \text{sonst.} \end{cases}$$

⊛ $4 = 2 \times 2$

Nachfolgend ist nun die rekursive Funktion dargestellt, die die Anzahl sortierter Ganzzahlen berechnen soll, welche mit `numberDigits` oder weniger Ziffern dargestellt werden können. Ein Aufruf von `NumberOfDigitSortedInts(3)` soll zum Beispiel die Anzahl sortierter Ganzzahlen zwischen 0 und 999 zurückgeben. Ergänzen Sie den Code entsprechend.

Below the recursive function is provided that shall compute the number of digit-sorted numbers that can be represented with `numberDigits` or less digits. A call to `NumDigitSortedInts(3)` should, for example, return the number of digit-sorted integers between 0 and 999. Complete the code accordingly.

```
int NumDigitSortedInts(const int numberDigits, const int leadingDigit) {
    int res = 0;
    if (numberDigits == 1)
        return (10-leadingDigit);
    else
        for (int i = leadingDigit; i < 10; ++i)
            res += NumDigitSortedInts(numberDigits-1,i);
    return res;
}

int NumDigitSortedInts(const int numberDigits) {
    return NumDigitSortedInts(numberDigits, 0);
}
```

⊛ 2

⊛ 2

Ende Aufgabe 4

5 Ringbuffer *(16 Punkte)*

Die Datenstruktur der Warteschlange (FIFO) wird üblicherweise mit einer verketteten Liste implementiert. Das hat unter anderem den Vorteil, dass darin beliebig viele Elemente aufgenommen werden können. Für jeden aufgenommenen Wert muss jedoch ein Listenelement alloziert werden.

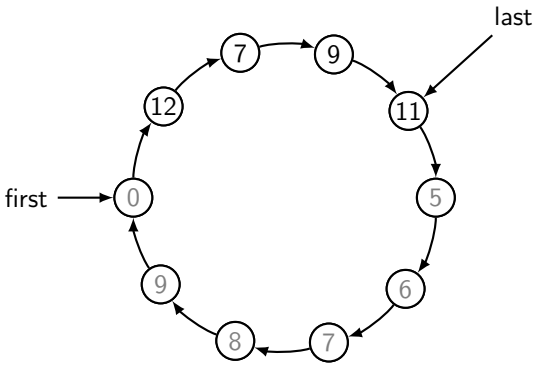
The queue data structure (FIFO) is usually implemented using a linked list. One of the advantages is that the queue can hold an arbitrary number of elements. For each entered value a list element has to be allocated, though.

Einmal allozierte, aber in der Queue nicht mehr genutzte Listenelemente sollen beim Einfügen wiederverwendet werden. Dazu kombinieren wir das Konzept eines Ringbuffers mit dem Konzept der Queue: die Warteschlange wird, wie üblich, mit einer einfach verketteten Liste implementiert. Wir ergänzen sie aber zu einer Ringstruktur. Wir nehmen an, dass einmal allozierte Listenelemente nie wieder freigegeben werden.

List elements that have been once allocated but are no longer used by the queue shall be reused for new values. In order to achieve this, we combine the concept of a ring-buffer with that of a queue: the queue is implemented as usual with a singly linked list but the list is connected to a ring structure. We assume that list elements once allocated will never be deallocated again.

Beispielszenario: / *Example scenario*

1. Erstelle neue Warteschlange / *Generate new queue*
2. $\underbrace{\texttt{Put}(1); \texttt{Put}(2); \ldots; \texttt{Put}(9)}$
   neun Aufrufe / *nine calls*
3. $\underbrace{\texttt{x} = \texttt{Get}(); \texttt{x} = \texttt{Get}(); \ldots; \texttt{x} = \texttt{Get}()}$
   neun Mal / *nine times*
4. `Put(12); Put(7); Put(9); Put(11);`



Betrachten Sie obiges Szenario, die dazugehörige Abbildung und den Code auf der nächsten Seite. Überlegen Sie sich zuerst, unter welchen Bedingungen der Buffer leer oder voll ist. Mit "voll" sei bezeichnet, dass das Einfügen eines zusätzlichen Wertes zu einer Allokation eines neuen Listenelementes führt. "Leer" ist der Buffer dann, wenn `Get` und `Put` gleich oft fehlerfrei aufgerufen wurden.

Consider the scenario above, the corresponding figure and the code on the next page. First think about conditions under which the buffer is empty or full. Here, "full" means that the insertion of a new value leads to an allocation of a new list element. "Empty" means that `Get` *and* `Put` *have been called without error equally often.*

a) Ergänzen Sie die Prozeduren `IsEmpty` und `IsFull` entsprechend.

Complete the code of procedures `IsEmpty` *and* `IsFull` *accordingly*

b) Vervollständigen Sie weiterhin unten stehende Implementation der Prozeduren `Put` und `Get`.

Complete the implementation of procedures `Put` *und* `Get` *below*

```
struct Node {
    Node* next; // pointer to the next item in the list
    int value;  // value of the current item

    Node (Node*const n, const int v) : next (n), value (v) {}
};

class Ring {
    Node* first; // pointer to the first item in the list
    Node* last;  // pointer to the last item in the list

public:

    Ring (){
        last = first = new Node(0, 0);
        last->next = first;
    }
    bool IsFull() const {
        return last->next == first;
    }

    bool IsEmpty() const{
        return first == last;
    }

    // Add a value to the queue. Enlarge the buffer only when necessary
    void Put (const int value) {
        if (IsFull()) {
            last->next = new Node(first, value);
        }
        last = last->next;
        last->value = value;
    }

    // Return the first element of the queue. Trap (Exception) if queue is empty.
    int Get() {
        assert(!IsEmpty());
        first = first -> next;
        return first->value;
    }

};
```

⊛ 4

⊛ 4

⊛ 4

⊛ 4

Ende Aufgabe 5

6 Lauflängenkodierung *(16 Punkte)*

Lauflängenkodierung ist ein einfaches Datenkompressionsverfahren, welches N aufeinander folgende identische Werte W durch ein Tupel (N, W) repräsentiert. Diese Methode findet zum Beispiel Anwendung in der Bildkompression. Beispiel:

Runlength encoding is a simple data compression technique that represents N consecutive identical values W by a tuple (N, W). This method is applied for image compression, for instance. Example:

$$\underbrace{25}_{(1,25)}, \underbrace{27}_{(1,27)}, \underbrace{26, 26, 26}_{(3,26)}, \underbrace{160}_{(1,160)}, \underbrace{175, 175}_{(2,175)}, \underbrace{255}_{(1,255)}$$

Im folgenden betrachten wir ein Datenarray mit vorzeichenlosen Byte-Werten (8-bit Wertebereich).

In the following we consider a data array with unsigned byte values (8-bit value domain).

Für Tupel mit Lauflänge $N = 1$ ist das Verfahren ineffizient, da ein einziger Wert durch zwei Bytes repräsentiert wird (Lauflänge und Wert). Um Platz zu gewinnen, können gewisse Tupel der Länge $N = 1$ als Einzelwert mit einem Byte dargestellt werden. Ob ein Byte ein Einzelwert darstellt oder das erste Byte eines Tupels ist, wird mit dem höchstwertigen Bit eines Bytes festgelegt. Ist dieses Bit eines Bytes gesetzt, so leitet das Byte in Tupel ein. Der Rest des Bytes (ohne das höchstwertige Bit) stellt die Lauflänge N dar und es folgt ein weiteres Byte mit dem Wert W. Falls das Bit nicht gesetzt ist, handelt es sich um einen Einzelwert.

For tuples with runlength $N = 1$ the method is inefficient because a single 8-bit value is represented by two bytes (Runlength and value). To gain some space, certain tuples of length $N = 1$ can be represented as single value in a byte. If a byte represents a single value or if it is the first byte of a tuple, is determined by the most significant bit of the byte. If this bit of a byte is set then the byte initiates a tuple. The rest of the byte (without most significant bit) represents the runlength N and a byte follows providing the value W. If the bit is not set it represents a single value.

a) Welches ist die maximal mögliche Lauflänge eines Tupels unter oben genannten Voraussetzungen? Ignorieren Sie ein mögliches Ausnutzen des ansonsten sinnlosen Falles $N = 0$. Antworten Sie mit einer Dezimalzahl.

What is the maximally possible runlength of a tuple under the conditions specified above? Ignore the possible optimisation of the otherwise useless case $N = 0$. Answer with a decimal number.

127

⊛ 3

b) Das Programm auf der folgenden Seite implementiert das Verfahren. Der Inhalt des Arrays `input` wird in den Vektor `output` komprimiert. Bestimmen Sie den Inhalt des Vektors `output` nach Ausführung der Funktion `main`. Die Werte sind dabei in dezimaler Representation anzugeben. Nicht verwendete Speicherstellen müssen mit "–" markiert werden.

The program on the following page implements the method. The content of the array `input` is compressed in vector `output`. Determine the content of vector `output` after execution of function `main`. The values have to be provided as decimal numbers. Unused memory locations have to be marked as "–".

⊛ 6

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
23	130	42	129	178	131	196	132	255	0	–	–

Programmcode für Aufgabe b) *Program code for b)*

```
#include <iostream>
#include <vector>

const int N = 12;
const int TupleFlag = 128;

const unsigned char input[N]
    = {23, 42, 42, 178, 196, 196, 196, 255, 255, 255, 255, 0};

std::vector<unsigned char> output;

void add_to_output(const int run_length, const int value) {
    if (run_length > 1 || value >= TupleFlag) {
        output.push_back(TupleFlag | run_length);
    }
    output.push_back(value);
}

int main() {
    int run_length = 1;
    for(int i = 1; i < N; ++i) {
        if (input[i] != input[i-1] || run_length == TupleFlag -1) {
            add_to_output(run_length, input[i - 1]);
            run_length = 1;
        } else {
            ++run_length;
        }
    }
    add_to_output(run_length, input[N -1]);

    std::cout << "output: ";
    for(unsigned int i = 0; i < output.size(); ++i) {
        std::cout << (int)output[i] << " ";
    }
    std::cout << "\n";
}
```

Fortsetzung der Aufgabe auf der nächsten Seite →
Problem continued on next page

c) Angenommen man möchte anstelle des Vektors `output` ein Array für die komprimierten Daten verwenden. Wie gross muss dieses Array definiert werden um einen beliebigen Input der Grösse N aufnehmen zu können?

Assume instead of vector `output` an array shall be used to store the compressed data. What minimal size is required for the array in order to be capable of storing an arbitrary input of length N?

2*N

⊛ 3

d) Für bestimmte Eingabedaten kann ein grösserer Bereich von Einzelwerten von Vorteil sein.

For certain input data a larger range of single values could be advantageous.

1. Wie kann man ein Byte markieren um Werte von $0 - 191$ als Einzelwerte darstellen zu können? Antworten Sie mit einem 8-bit Wert in binärer Schreibweise.

   How can a byte be marked in order to be capable of representing values from $0 - 191$ as single values? Answer with an 8-bit value in binary representation.

   11000000

   ⊛ 2

2. Wie gross wäre dann eine maximale Lauflänge?

   What would then be the maximal run length?

   63

   ⊛ 2

Ende Aufgabe 6