

Name, Vorname: .....

Legi-Nummer: .....

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen 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:**

**General guidelines:**

1. Dauer der Prüfung: 60 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für gesprochene Sprachen).
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.
4. Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben (und direkt darunter, falls mehr Platz benötigt wird). Ungültige Lösungen bitte deutlich durchstreichen! Korrekturen bei Multiple-Choice Aufgaben unmissverständlich anbringen!
5. Falls Sie sich durch irgendjemanden oder irgendetwas gestört fühlen, melden Sie dies sofort der Aufsichtsperson.
6. 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.
7. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen. Es darf zur gleichen Zeit immer nur eine Studentin oder ein Student zur Toilette.
8. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

- Exam duration: 60 minutes.*
- Permitted examination aids: dictionary (for spoken languages).*
- 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 (and directly below, if more space is needed). 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  | 7  | 8  | Σ  |
| Punkte  |     |     |    |    |   |    |    |    |    |
| Maximum | 7.5 | 7.5 | 12 | 10 | 6 | 10 | 12 | 10 | 75 |

# 1 Typen und Werte (Basistypen) (7.5 Punkte)

Geben Sie für jeden der Ausdrücke auf der rechten Seite jeweils C++-Typ (0.5 P) und Wert (1 P) an! Wenn der Wert nicht bestimmt werden kann, schreiben Sie "undefiniert".

Die verwendeten Variablen sind wie folgt deklariert / initialisiert.

---

```
int n = 9;
int f = 5;
int d = 3;
unsigned int u = 8;
int x;
int y;
bool a = x > 0;
bool b = y < 0;
```

---

*For each of the expressions on the right, provide the C++ type (0.5 P) and value (1 P)!*

*The used variables have been declared / initialized as shown above. If a value cannot be determined, write "undefined".*

---

(a) `n / f * d`

1.5 P

**Typ/Type**

**Wert/Value**

---

(b) `-n - --f`

1.5 P

**Typ/Type**

**Wert/Value**

---

(c) `-0x12 * 0xa`

1.5 P

**Typ/Type**

**Wert/Value**

---

(d) `u-9 == -1`

1.5 P

**Typ/Type**

**Wert/Value**

---

(e) `a || !b && !a || b`

1.5 P

**Typ/Type**

**Wert/Value**

## 2 Typen und Werte (Structs und Pointers) (7.5 Punkte)

Geben Sie für jeden der Ausdrücke auf der rechten Seite jeweils C++-Typ (0.5 P) und Wert (1 P) an!

Die verwendeten Variablen seien deklariert und initialisiert wie folgt.

---

```
struct S {
    int v;
    S* n;

    S(int V, S* N): v(V), n(N)
        {}

    S(): v(5), n(0)
        {}
};

S s;
S* ps = new S(10, 0);
S* pt = new S(9, ps);

int a[2] = {1,2};
int *x = &a[0];

double d[2][2] = {{1,0},{3,4}};
```

---

*For each of the expressions on the right, provide the C++ type (0.5 P) and value (1 P)!*

*The used variables have been declared and initialized as shown above.*

---

(a) `s.n` 1.5 P

**Typ/Type**

**Wert/Value**

---

(b) `ps->v * pt->n->v` 1.5 P

**Typ/Type**

**Wert/Value**

---

(c) `a[*x] * x[*a]` 1.5 P

**Typ/Type**

**Wert/Value**

---

(d) `d[0][0] * d[1][1]` 1.5 P

**Typ/Type**

**Wert/Value**

---

(e) `&d[0][0] < &d[0][1]` 1.5 P

**Typ/Type**

**Wert/Value**

### 3 Programmausgaben (12 Punkte)

Betrachten Sie folgendes Programm. Beantworten Sie die Fragen auf der rechten Seite.

---

```
#include <iostream>

void func_a(char* start, char* end) {
    for(char* it = start; it < end; ++it)
        *it = *it+1;
}

void func_b(char* start, char* end) {
    for(char* it = start; it < end; it+=2)
        std::cout << *it << " ";
}

void func_c(int b) {
    if (b > 0){
        func_c(b/2);
        if (b % 2 == 1)
            std::cout << "X";
        else
            std::cout << "-";
    }
}

int func_d(int* a, int& i) {
    std::cout << a[i];
    i = a[i];
    return i;
}

int main() {
    // insert code from right hand side here
    return 0;
}
```

---

*Consider the program above. Answer the questions on the right hand side.*

Was gibt das Programm der linken Seite jeweils aus, wenn man folgende Codestücke in die main-Funktion einfügt. *What is each output of the program on the left hand side when the following code pieces are inserted in the main function.*

---

(a) `unsigned int x = 7;`  
`do {`  
    `std::cout << x << " ";`  
    `x = (2 * x + 1) % 17;`  
`} while (x != 0);`

3 P

---

(b) `char src[] = { 'd', 'c', 's', 'b', 'h', 'l' };`  
`func_a(src, src+3);`  
`func_b(src, src+6);`

3 P

---

(c) `func_c(22);`

3 P

---

(d) `int i = 0;`  
`int test[] = {2,4,1,0,3};`  
`while (func_d(test,i) > 0);`

3 P

## 4 Normalisiertes Fließkommasystem (10 Punkte)

Wir betrachten das unten angegebene normalisierte Fließkommazahlensystem  $F^*$ . Beantworten Sie die Fragen auf der rechten Seite!

Anmerkung: Falls nötig runden Sie binär arithmetisch, d.h., eine 1 wird aufgerundet, eine 0 wird abgerundet.

Beispiel: in  $F^*$  wird die binäre dargestellte Zahl  $1.01\underline{0}..$  zu  $1.01$  abgerundet, während  $1.01\underline{1}..$  zu  $1.10$  aufgerundet wird.

---

$F^*$  ( $\beta, p, e_{\min}, e_{\max}$ ) mit / with

$$\beta = 2$$

$$p = 3$$

$$e_{\min} = -4$$

$$e_{\max} = 4$$

---

*Consider the normalized floating point number system  $F^*$  defined above. Answer the questions on the right side!*

*Note: If necessary use binary arithmetic rounding, i.e., round up for a 1 and down for a 0.*

*Example: in  $F^*$  the binary represented number  $1.01\underline{0}..$  is rounded down to  $1.01$ , while  $1.01\underline{1}..$  is rounded up to  $1.10$ .*

- (a) Wie viele unterschiedliche positive Werte beinhaltet das normalisierte Fließkommasytem  $F^*$ ? 2 P  
*How many different positive values can be represented using the normalized floating point system  $F^*$ ?*

- (b) Geben Sie die grösste Zahl und die kleinste positive Zahl, die das normalisierte Fließkommasytem  $F^*$  repräsentieren kann, an. Beide Antworten sind in **dezimaler Darstellung** anzugeben. 2 P  
*Provide the largest number and the smallest positive number representable by the normalized floating point system  $F^*$ . Provide both answers in **decimal representation**.*

grösste Zahl / *largest number*

kleinste positive Zahl / *smallest positive number*

- (c) Berechnen Sie folgende Ausdrücke so wie sie geklammert sind, indem Sie gemäss der Regeln für das Rechnen mit Fließkommazahlen jedes Zwischenresultat (und das Endresultat) im gegebenen Fließkommazahlensystem darstellen. Vergessen Sie das korrekte Runden nicht. 6 P  
*Compute the following expressions as the parentheses suggest, representing each intermediate result (and the final result) in the given normalized floating point system according to the rules of computing with floating point numbers. Do not forget to round correctly.*

| $(10 + 0.5) + 0.5$   |                       | $(0.5 + 0.5) + 10$   |                       |
|--|-----------------------|--|-----------------------|
| dezimal  | binär / <i>binary</i> | dezimal  | binär / <i>binary</i> |
| 10   |                       | 0.5  |                       |
| + 0.5  |                       | + 0.5  |                       |
| =  |                       | =  |                       |
| + 0.5  |                       | + 10   |                       |
| = <span style="border: 1px solid black; display: inline-block; width: 40px; height: 20px; vertical-align: middle;"></span> ← |                       | = <span style="border: 1px solid black; display: inline-block; width: 40px; height: 20px; vertical-align: middle;"></span> ← |                       |

## 5 BNF I (6 Punkte)

Die folgende BNF definiert eine Sprache zur Beschreibung von kantengewichteten Bäumen. Ein geklammerter Ausdruck steht für einen Baum, ganze Zahlen bezeichnen Kantengewichte. Ein illustrierendes Beispiel ist unter der BNF zu finden.

Beantworten Sie die Fragen auf der rechten Seite!

*The BNF displayed below defines a language for trees with weighted edges. An expression in parentheses stands for a tree, integer numbers designate edge weights. An illustrating example is displayed below the BNF.*

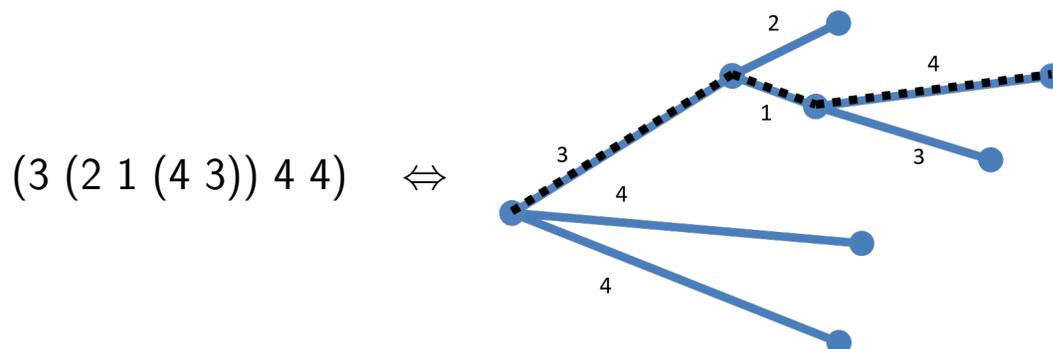
*Answer the questions on the right side!*

---

```
Tree = '(' Branches ')'.  
Branches = Branch | Branch Branches.  
Branch = Length | Length Tree.  
Length = unsigned int.
```

---

Beispiel / *Example:*



Die gestrichelten Linien kennzeichnen Pfad mit maximaler Länge. Das ist nur für Aufgabe 6 relevant.

*The dashed lines designate the path with maximal length. This is only relevant for problem 6.*

---

(a) Wahr oder falsch? *true or false?*

1 P

Die folgende Zeichenkette ist ein gültiger Baum (Tree) nach der BNF:

*The following string is a valid tree (Tree) according to the BNF:*

(10 20 30)

---

(b) Wahr oder falsch? *true or false?*

1 P

Die folgende Zeichenkette ist ein gültiger Baum (Tree) nach der BNF:

*The following string is a valid tree (Tree) according to the BNF:*

()

---

(c) Wahr oder falsch? *true or false?*

1 P

Die folgende Zeichenkette ist ein gültiger Baum (Tree) nach der BNF:

*The following string is a valid tree (Tree) according to the BNF:*

(10 5 (20 30 40) 30 40 (5 2 1) 45)

---

(d) Wahr oder falsch? *true or false?*

1 P

Die folgende Zeichenkette ist ein gültiger Baum (Tree) nach der BNF:

*The following string is a valid tree (Tree) according to the BNF:*

10 (10 20 30)

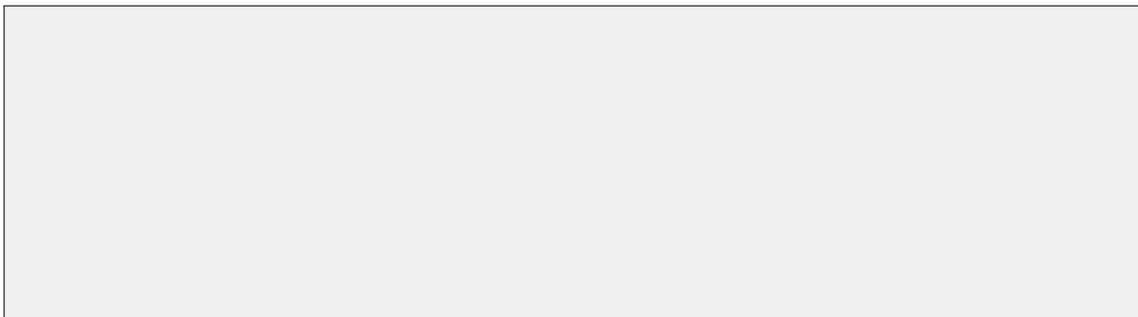
---

(e) Zeichnen Sie den Baum, welcher zu folgender Eingabe gehört.

2 P

*Draw the tree that belongs to the following input.*

(1 (2 (3) 4) 5)



## 6 BNF II (10 Punkte)

Betrachten Sie folgenden Code und beantworten Sie die Fragen auf der rechten Seite.

*Consider the following code and answer the questions on the right hand side.*

---

```
// POST: leading whitespace characters are extracted from is, and the
//      first non-whitespace character is returned (0 if there is none)
char lookahead (std::istream& is);

// Tree = '(' Branches ')'.
// Returns length of longest outgoing path.
int Tree (std::istream& is){
    char c;
    is >> c; length = ; is >> c;
    return length;
}

// Branches = Branch | Branch Branches.
int Branches (std::istream& is){
    int length = Branch(is);
    if (lookahead(is) != ')'){
        int bLength = Branches(is);
        
    }
    return length;
}

// Branch = Length | Length Tree.
int Branch(std::istream& is){
    int length = Length(is);
    if (lookahead(is) == '(') return ;
    return length;
}

// Length = unsigned int.
// Returns integer present at input stream
int Length (std::istream& is);
```

---

Die folgende main-Funktion soll zu einem Baum, welcher gemäss der BNF der vorigen Aufgabe am Eingabestrom vorliegt, die maximale Pfadlänge ausgeben. Die Länge eines Pfades ist die Summe der Gewichte entlang des Pfades. Ein Pfad eines Baumes geht jeweils von der Wurzel des Baumes bis zu einem Blatt. Im Beispiel auf Seite 10 ist der Pfad mit maximaler Länge  $3+1+4=8$  gestrichelt eingezeichnet.

*The following main function shall return the maximal path length of a tree that is provided at the input stream according to the BNF defined in the previous task. The length of a path is given by the sum of its weights. A path of a tree starts from the tree root and ends at a leaf. In the example on page 10 the path with maximal length  $3+1+4=8$  is dashed.*

---

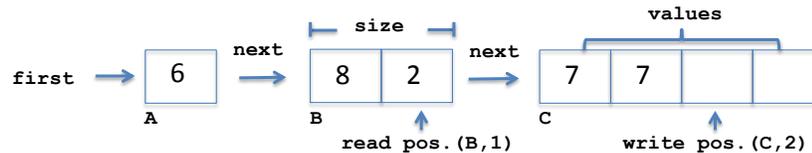
```
int main() {  
    int length = Tree(std::cin);  
    std::cout << "Maximal path length= " << length << "\n";  
    return 0;  
}
```

---

- 
- (a) Vervollständigen Sie den Code der Funktion Tree entsprechend. 3 P  
*Complement the code of function Tree accordingly.*
- 
- (b) Vervollständigen Sie den Code der Funktion Branches entsprechend. 4 P  
*Complement the code of function Branches accordingly.*
- 
- (c) Vervollständigen Sie den Code der Funktion Branch entsprechend. 3 P  
*Complement the code of function Branch accordingly.*

## 7 Verkettete Liste von Arrays (12 Punkte)

Die Datenstruktur `AppendList` implementiert eine Liste von Elementen mittels verketteten Arrays. Jedes Array ist doppelt so gross wie das vorherige, daher muss nicht für jedes Listenelement neuer Speicher alloziert werden. Beantworten Sie die Fragen auf der rechten Seite.



*The data structure `AppendList` implements a list of elements using linked arrays. Each array has twice the size of the previous array, thus avoiding memory allocation for each list element. Answer the questions on the right hand side.*

```
struct Node {
    int* values; int size; Node* next;

    Node(int s) : values(new int[s]), size(s), next(0) { }
    ~Node() { delete[] values; }
};

class AppendList {
    Node* first;
    Node* write_node; int write_idx; // write position
    Node* read_node; int read_idx; // read position
public:
    AppendList() : read_idx(0), write_idx(0), first(new Node(1))
                 , read_node(first), write_node(first) { }
    void write(int value) { // adds element at end of list
        if (write_idx == write_node->size) {
            write_node->next = new Node(2 * write_node->size);
            write_node = write_node->next; write_idx = 0;
        }
        write_node->values[write_idx++] = value;
    }
    int read() { // returns element at current read position
        if (read_idx == read_node->size) {
            read_node = read_node->next; read_idx = 0;
        }
        return read_node->values[read_idx++];
    }
};
```

- 
- (a) Wieviele Instanzen der Klasse Node enthält eine initial leere AppendList, nachdem ihre Funktion write N mal aufgerufen wurde? 2 P

*How many instances of class Node does an initially empty AppendList contain after its function write has been called N times?*

N = 7:

N = 23:

- 
- (b) Vervollständigen Sie den Destruktor AppendList::~~AppendList(), so dass er den allozierten Speicher freigibt: 4 P

*Complete the destructor AppendList::~~AppendList() to free all allocated memory:*

---

```
AppendList::~~AppendList() {  
    Node* node = first;  
    while (node != ) {  
        Node* tmp = ;  
        node = ;  
        delete ;  
    }  
}
```

---

- (c) Vervollständigen Sie die Funktion AppendList::~reset, welche die Leseposition an das vorderste Element der Liste zurücksetzt. 3 P

*Complete the function AppendList::~reset that resets the read position to the element at the begin of the list.*

---

```
void AppendList::~reset() {  
    ;  
    ;  
}
```

---

- (d) Vervollständigen Sie die Funktion AppendList::~available, welche true zurückgibt, sofern weitere Elemente von der Liste gelesen werden können (Leseposition vor Schreibposition). 3 P

*Complete the function AppendList::~available that returns true if more elements can be read from the list (read position before write position).*

---

```
bool AppendList::~available() {  
    return ;  
}
```

## 8 Bilder mit Tiefe (10 Punkte)

Betrachten Sie folgenden Code und beantworten Sie die Fragen auf der rechten Seite. *Consider the following code and answer the questions on the right hand side.*

---

```
#include <cassert>
bool check_sizes(const int* begin1, const int* end1,
                const int* begin2, const int* end2) {
    return ;
}

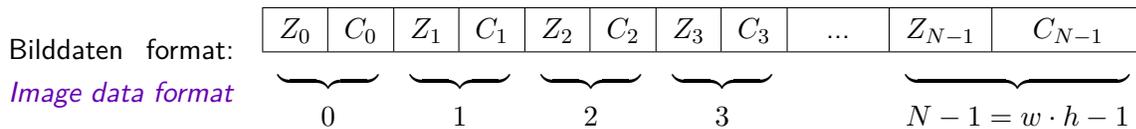
// POST: merge source image defined by begin2, end2
//       into destination image defined by begin1, end1
void merge(int* begin1, int* end1, const int* begin2, const int* end2) {
    assert(same_size(begin1, end1, begin2, end2));
    int* p1 = begin1; // initialize pointers
    int* p2 = begin2;
    while(p1 != end1) {
        // check and overwrite pixels of destination image if required
        
        // increment both pointers to next pixel
        p1 += ; p2 += ;
    }
}

int main() {
    int image1[] = { 1000, 0, 1000, 0, 1000, 0, 1000, 0 // row 1
                   ,1000, 0, 10,   5, 1000, 0, 1000, 0 // row 2
                   ,10,   5, 12,   5, 12,   5, 1000, 0}; // row 3
    int image2[] = { 1000, 0, 1000, 0, 1000, 0, 1000, 0 // row 1
                   ,5   , 3, 10,   3, 15,   3, 16,   3 // row 2
                   ,5   , 4, 10,   4, 15,   4, 16,   4}; // row 3
    merge();
    return 0;
}
```

---

Die Funktion `merge` fügt anhand von Raamtiefeninformation das Bild  $I_2$  in das Bild  $I_1$  ein. Bilder sind in eindimensionalen Arrays von Bildpunkten abgelegt. Jeder Bildpunkt wird durch ein Paar  $(Z, C)$  repräsentiert.  $Z$  und  $C$  sind beides positive Integerwerte.  $Z$  ist die Tiefe im Raum,  $C$  ist die Farbe. Der Betrachter steht bei Tiefe 0. Je höher  $Z$ , desto weiter weg liegt ein Bildpunkt vom Betrachter. Die Funktion `merge` überschreibt einen Bildpunkt im Bild  $I_1$ , falls der entsprechende Bildpunkt in  $I_2$  näher bei Betrachter ist, also einen tieferen  $Z$  Wert hat, als derjenige von  $I_1$ .

*The function `merge` inserts the image  $I_2$  into the image  $I_1$  using spatial depth information. Image are stored in one dimensional arrays of pixels. Each pixel is represented by the pair  $(Z, C)$ .  $Z$  and  $C$  are both positive integer values.  $Z$  represents the spatial depth,  $C$  represents the color. The observer is located at depth 0. The higher its depth  $Z$  is the further away is a pixel from the observer. The function `merge` overwrites a pixel in image  $I_1$  if the corresponding pixel in  $I_2$  is closer at the observer, i.e., has a lower value of  $Z$  than the one of  $I_1$ .*



- (a) Vervollständigen Sie die Funktion `check_sizes`, so dass sie `true` zurückgibt, falls beide Bilder gleich gross sind und jeweils eine geradzahlige Anzahl von ints enthalten. 2 P
- Complete function `check_sizes`, such that it returns true if both images are of the same size and contain an even number of ints each.*

- (b) Vervollständigen Sie den Aufruf der Funktion `merge` in `main`, so dass die Bilder `image1` und `image2` vollständig an `merge` übergeben werden. Bildpunkte von `image1` sollen ggfs. durch Bildpunkte von `image2` ersetzt werden. 2 P
- Complete the call to `merge` in the function `main()`, such that that the entire content of both images `image1` and `image2` are passed to `merge`. Pixels of `image1` shall be replaced by pixels of `image2` if applicable.*

- (c) Vervollständigen Sie die Funktion `merge` entsprechend. 6 P
- Complement the function `merge` accordingly.*