Aufgabe 1: Typen und Werte (Basistypen) (6P)

Geben Sie für jeden der Ausdrücke auf der nächsten Seite jeweils C++Typ und Wert an. Wenn der Wert nicht bestimmt werden kann, schreiben Sie "undefiniert".
Die verwendeten Variablen sind wie folgt deklariert / initialisiert.

For each of the expressions on the next page provide the C++type and value. If a value cannot be determined, write "undefined".
The used variables are declared as follows.

```
int i = 10;
double d = 2;
int m = 7;
int pre = 1;
int post = 1;
unsigned int u = 0;
```

(a) `1 / 4 * d` /1P

Typ / *Type*	Wert / *Value*
`double`	`0`

(b) `m % m` /1P

Typ / *Type*	Wert / *Value*
`int`	`0`

(c) `i = 10` /1P

Typ / *Type*	Wert / *Value*
`int`	`10`

(d) `i += i` /1P

Typ / *Type*	Wert / *Value*
`int`	`20`

(e) `++pre / post--` /1P

Typ / *Type*	Wert / *Value*
`int`	`2`

(f) `u-5 < 6` /1P

Typ / *Type*	Wert / *Value*
`bool`	`false`

9 Normalisiertes Fliesskommasystem *(6 Punkte)*

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

Anmerkung: Falls nötig runden Sie arithmerisch, d.h., eine 1 wird aufgerundet, eine 0 wird abgerundet. Z.b. $1.0101\underline{0}$ wird in F^* zu 1.0101 abgerundent, während $1.0101\underline{1}$ in F^* zu 1.0110 aufgerundet wird.

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

$\beta = 2$

$p = 5$

$e_{\min} = -2$

$e_{\max} = 2$

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^*): $1.0101\underline{0}$ is rounded down to 1.0101, while $1.0101\underline{1}$ is rounded up to 1.0110.

(a) Wie viele unterschiedliche positive Werte beinhaltet das normalisierte Fliesskommasystem F^*? 1 P

How many different positive values can be represented using the normalized floating point system F^*?

80

(b) Geben Sie die grösste Zahl und die maximale Präzision (kleinste positive Zahl), die das normalisierte Fliesskommasystem F^* repräsentieren kann, an. Beide Antworten sind in **dezimaler Darstellung** anzugeben. 2 P

Provide the largest number and the highest precision (smallest positive number) representable by the normalized floating point system F^*. Provide the answers in **decimal representation**.

grösste Zahl / largest number: 7.75

maximale Präzision / highest precision: 0.25

(c) Die Dezimalzahl $x = 2.5625$ ist im normalisierten Fliesskommasystem F^* nicht darstellbar. Geben Sie die Zahl $\hat{x}$ an, welche in F^* enthalten ist, und x am nächsten liegt. Geben Sie die Antwort in **binärer Darstellung** an. Hinweis: Verwenden Sie arithmetisches Runden. 3 P

The decimal number $x = 2.5625$ is not representable in the normalized floating point system F^*. Provide the number $\hat{x}$ that is closest to x and is representable in F^*. Provide the answer in **binary representation**. Hint: Use arithmetic rounding to find this number.

$\hat{x} =$ $(10.1001 \rightarrow)10.101 = 1.0101 \cdot 2^1$

Berechnen Sie den absoluten Rundungsfehler $|\hat{x} - x|$. Geben Sie die Antwort in **dezimaler Darstellung** an.

Calculate the absolute rounding error $|\hat{x} - x|$. Provide the answer in **decimal representation**.

$|\hat{x} - x| =$ 0.0625

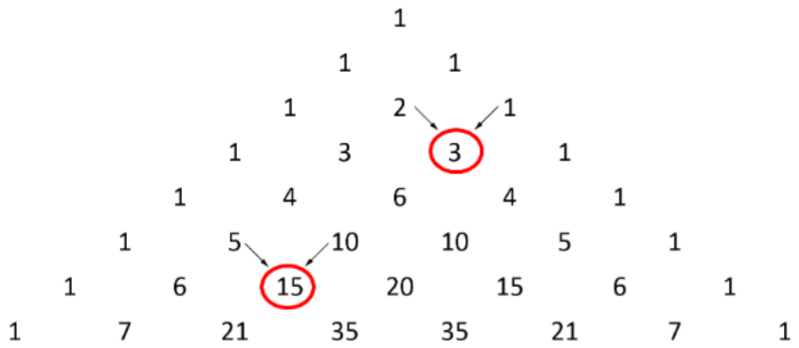
Aufgabe 5: Rekursion: Pascalsches Dreieck (13P)

Das Ziel dieser Aufgabe ist, die Zahl einer gegebene Position im Pascalschen Dreieck zu berechnen (siehe Bild). Verfolständigen Sie das gegebene Programm entsprechend.
Eine Zahl an einer gegebenen Position innerhalb des Dreiecks entspricht der Summe der beiden Zahlen links und rechts in der darüberliegenden Zeile. Alle Zahlen am linken und rechten Rand des Dreiecks haben den Wert 1 gesetzt.
Beispiel: Die eingekreiste 15 im untenstehenden Dreieck steht in der $7ten$ Zeile, an $3te$ Stelle. Gleichermassen: Die eingekreiste 3 steht in der $4ten$ Zeile, an $3ter$ Stelle.

In this task, you need to complete the program below to compute the value of a given position in Pascal's triangle (see image).
A number in a given position within the triangle corresponds to the sum of the two numbers on the left and right of the row above. All numbers on the left and right edge of the triangle are set to have value 1.
Example: The encircled 15 in the below triangle corresponds to the 7^{th} row, on the 3^{rd} position. Similarly, encircled 3 corresponds to the 4^{rd} row, on the 3^{rd} position.

Das Programm soll den Nutzer auffordern, eine Zeile und die Stelle in dieser Zeile einzugeben. Dabei wird **mit 1 angefangen** zu zählen. Ausserdem soll das Programm vor der Berechnung überprüfen, ob die eingegebene Positionskombination gültig ist.

*The program should prompt the user to input a row, and a position in that row, considering **1-based** counting.*
The program should ensure that the input is valid before computing a value for the position.



/3P (a) Vervollständigen Sie die Funktion `main`.

Complete the function `main`.

/5P (b) Vervollständigen Sie die Funktion `check_input_validity`. Sie können annehmen, dass die eingegebenen Variablen immer vom richtigen Datentyp sind.

Complete the function `check_input_validity`. *You can assume that the user always enters integer values as the input variables.*

/5P (c) Vervollständigen Sie die Funktion `compute_pascal` mit Hilfe von **Rekursion**.

Complete the function `compute_pascal` *by using **recursion**.*

```
#include <iostream>
#include <cassert>

bool check_input_validity(int row, int pos){
    if (row < pos || pos < 1){
        std::cout << "Invalid: no element at the given row-position pair.";
        return false;
    }
    else {
        return true;
    }
}

int compute_pascal(int row, int pos){
    if (pos == 1){
        return 1;
    }
    else if (pos == row){
        return 1;
    }
    else {
        return compute_pascal(row-1, pos) + compute_pascal(row-1, pos-1);
    }
}

void main(){

    //Input
    std:: cout<< "Please input a row and a position along the row: ";

    int row, pos;
    std::cin >> row >> pos;

    //Check whether the input integers correspond to a valid entry
     assert (check_input_validity(row, pos));

    //Display the result
    std::cout << "The value at row " << row << " and position " << pos << " is "
              << compute_pascal(row, pos);
}
```

Aufgabe 4: EBNF I (6P)

Die folgende EBNF definiert erlaubte Anweisungen einer vereinfachten Programmiersprache.
Beantworten Sie die Fragen auf der rechten Seite.
Anmerkung: Leerschläge sind im Rahmen der EBNF bedeutungslos.

The following EBNF defines the allowed statements of a programming language. Answer the questions on the right hand side.

Remark: *Whitespaces are irrelevant in the context of this EBNF.*

```
Statement      = Expression ';'.
Expression     = Simple [':' Simple].
Simple         = Designator | Integer.
Designator     = Identifier {'.' Identifier | '(' [ExpressionList] ')'}.
ExpressionList = Expression { ',' Expression }.

Integer        = Digit {Digit}.
Digit          = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .

Identifier     = Letter {Letter}.
Letter         = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' .
```

(a) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF: *The following string corresponds to a valid statement according to the EBNF:* /1P

```
a(2:5);
```

√ wahr / *true* ◯ falsch / *false*

(b) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF: *The following string corresponds to a valid statement according to the EBNF:* /1P

```
a(b).c:d;
```

√ wahr / *true* ◯ falsch / *false*

(c) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF: *The following string corresponds to a valid statement according to the EBNF:* /1P

```
a(3).3;
```

◯ wahr / *true* √ falsch / *false*

(d) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF: *The following string corresponds to a valid statement according to the EBNF:* /1P

```
a(3):3;
```

√ wahr / *true* ◯ falsch / *false*

(e) Ändern Sie genau eine Produktionsregel der EBNF ab, so dass folgende Anweisung (statement) gültig wird. *Modify one and only one production rule of the EBNF such that the following statement becomes valid.* /2P

```
a3(c3):a4(c4);
```

Geänderte Zeile / *Modified line*:

```
Identifier = Letter {Letter | Digit}.
```

Aufgabe 5: EBNF II (8P)

Auf der nächsten Seite ist Code abgebildet, mit welchem identifiziert werden soll, ob eine Zeichenkette eine im Sinne obiger EBNF gültige Anweisung (Statement) darstellt. Folgender Code, welcher nur als Funktionsdeklaration vorliegt, soll als korrekt implementiert vorausgesetzt werden.

Consider the code on the next page that shall be used to check if a string provides a valid statement corresponding to the EBNF above. The following code that is only provided as a function declaration can be considered correctly implemented.

```
// POST: when the next available non-whitespace character equals c,
//       it is consumed and the function returns true,
//       otherwise the function returns false.
bool has(std::istream& is, char c);

// Integer = Digit {Digit}.
bool Integer (std::istream& is);

// Identifier = Letter {Letter}.
bool Identifier (std::istream& is);
```

/2P (a) Die Funktion `Expression` wird im Code anscheinend zweimal deklariert. Erklären Sie warum.

The function `Expression` seems to be declared twice in the code. Explain why.

Es gibt eine zirkuläre Abhängigkeit zwischen den Funktionen. Daher muss mindestens eine Funktion vor ihrer Definition deklariert werden. Sonst wäre sie in mindestens einer anderen Funktion nicht sichtbar.

/2P (b) Ergänzen Sie die Funktion `Designator`, so dass Sie die entsprechende EBNF-Zeile korrekt implementiert.

Complement function `Designator` such that it implements the corresponding EBNF line correctly.

/2P (c) Ergänzen Sie die Funktion `ExpressionList`, so dass Sie die entsprechende EBNF-Zeile korrekt implementiert.

Complement function `ExpressionList` such that it implements the corresponding EBNF line correctly.

/2P (d) Ergänzen Sie die Funktion `Expression`, so dass Sie die entsprechende EBNF-Zeile korrekt implementiert.

Complement function `Expression` such that it implements the corresponding EBNF line correctly.

```
bool Expression (std::istream& is);
// ExpressionList = Expression { ',' Expression }.
bool ExpressionList(std::istream& is){
    if (!Expression(is)) return false;
    while (has(is, ',')){
        if (!Expression(is)) return false;
    }
    return true;
}
// Designator = Identifier {'.' Identifier | '(' [ExpressionList] ')'}.
bool Designator(std::istream& is){
    if (!Identifier(is)) return false;
    while(true){
        if (has(is, '.')){
            if (!Identifier(is)) return false;
        } else if (has(is, '(')){
            bool ignore = ExpressionList(is);
            if (!has(is,')')) return false;
        } else
            return true;
    }
}
// Simple = Designator | Integer.
bool Simple(std::istream& is){
    return Integer(is) || Designator(is);
}
// Expression = Simple [':' Simple].
bool Expression(std::istream& is){
    if (!Simple(is)) return false;
    return !has(is,':') || Simple(is);
}
// Statement = Expression ';'.
bool Statement(std::istream& is){
    return Expression(is) && has(is, ';');
}
```

Aufgabe 5: Klassen und Operator Overloading (10P)

Auf der rechten Seite sehen Sie die Implementation einer Klasse für Zeit. Unten auf dieser Seite sehen Sie, wie die Klasse benutzt werden soll. Implementieren Sie die fehlenden Teile wie im folgenden beschrieben.

On the right hand side you see the implementation of a class for time. Below on this page you see how the class should be used. Implement the missing parts as described in the following.

/3P (a) Implementieren Sie den Operator == so, dass er true zurückgibt, wenn zwei Zeiten gleich sind. Gleich sind hier zwei Zeiten, wenn sie die gleiche Tageszeit repräsentieren. In diesem Sinne sind die Zeiten 24:05:59 und 0:05:59 gleich.

Implement the operator == such that it returns true when the two times equal. Two times are equal when they represent the same time of the day. In this sense, 24:05:59 equals 0:05:59.

/1P (b) Implementieren Sie den Operator != so, dass er genau dann true zurückgibt, wenn die zwei Zeiten nicht gleich sind.

Implement the operator != such that it returns true if and only if the times to be compared are not equal.

/3P (c) Implementieren Sie den arithmetischen Zuweisungsoperator += so, dass Zeilen 5 und 6 des Codes der Main-Funktion erwartungsgemäss funktionieren

Implement the arithmetic assignment operator += such that lines 5 and 6 of the main function below works as expected.

/3P (d) Implementieren Sie den Ausgabeoperator << so, dass Zeile 7 des Codes der Main-Funktion erwartungsgemäss funktioniert.

Implement the output operator << such that line 7 of the code of the main function works as expected.

```
1 int main(){
2     Time second (0,0,1);
3     Time examStart (14,04,58);
4     Time examDuration (01,02,05);
5     Time examEnd = examStart + examDuration;
6     for (Time t=examStart; t != examEnd; t += second){
7         std::cout << t << std::endl;
8     }
9     // output:
10    // 14:4:58
11    // 14:4:59
12    // ...
13    // 15:7:2
14 }
```

```
class Time{
    int sec; // total number of seconds

    public:
    // construct time from hours, minutes and seconds
    Time (int h, int m, int s): sec(s+m*60+h*3600){}
    // add t to this time
    Time& [[1p]] operator+= (const Time& t){
        sec += t.sec; [[1p]]
        return *this; [[1p]]
    }
    // two times match when they represent the same time during a day
    // Example: 0:0:5 must match 24:0:5
    bool operator== (const Time& t) const [[2p]] {
        return sec % (3600*24) == t.sec % (3600*24); [[1p]]
    ;
    }
    // write time to stream out
    void write (std::ostream& out) const {
        out << sec / 3600 % 24 << ":" << sec / 60 % 60 << ":" << sec % 60;
    }
};
// output time on stream o
std::ostream& [[1p]] operator<< (std::ostream& o, const Time t){
    t.write(o); [[1p]]
    return o; [[1p]]
}
// return if times are not equal
bool operator !=(const Time& t1, const Time& t2){
    return !(t1 == t2)
;
}
// add time t1 and time t2
Time operator+(Time t1, const Time & t2){
    return t1 += t2;
}
```

Aufgabe 2: Konstrukte (16P)

Geben Sie zu folgenden Codestücken jeweils die erzeugte Ausgabe an.

Provide the output for each of the following pieces of code.

/4P (a)

```
int a[6] = {1, 3, 5, 7, 9, 11};
int* x = a;
int& k = *x;
x++;
std::cout << *(x+k);
```

Ausgabe / *Output*: 5

/4P (b)

```
float p[6] = {0.5, 1.0, 1.5, 2.0, 0.5, 0.0};
float* s = &p[4];
float* r = &p[2];
std::cout << (r[2] + *(s+1))
```

Ausgabe / *Output*: 0.5

/4P (c)

```
void my_function(int* m) {
    *m = 2;
}
int values[] = {1, 3, 5};
int* v = values;
my_function(v++);
std::cout << v[0];
```

Ausgabe / *Output*: 3

/4P (d)

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

Node s = Node(10, nullptr);
Node t = Node(20, &s);
s.next = &t;
std::cout << (s.next->next->value);
```

Ausgabe / *Output*: 10