Geben Sie für jeden der drei Ausdrücke auf der rechten Seite jeweils C++-Typ und Wert an.

Das Array a sei deklariert und initialisiert wie folgt.

```
double a[] = {2.33, 0.25, 2.33, 1.0};
```

For each of the 3 expressions on the right, provide the C++ type and value!

Array a has been declared and initialized as shown above.

- `1.25 / a[1] == 5 && false`

  Typ/Type: `bool`

- `1.25 / a[1] == 5 && false`

  Wert/Value: `false`

- `a < &a[2]`

  Typ/Type: `bool`

- `a < &a[2]`

  Wert/Value: `true`

- `0.375e3 + *(a + 3) + a[1]`

  Typ/Type: `double`

- `0.375e3 + *(a + 3) + a[1]`

  Wert/Value: `376.25`

Geben Sie für jeden der drei Ausdrücke auf der rechten Seite jeweils C++-Typ und Wert an!

Variablen `x` und `y` seien deklariert und initialisiert wie folgt.

```
int x = 9;
int y = 4;
```

For each of the 3 expressions on the right, provide the C++ type and value!

Variables `x` and `y` have been initialized as shown above.

- `85 / x % y < 5`

  Typ/Type: `bool`

- `85 / x % y < 5`

  Wert/Value: `true`

- `x % y / 2`

  Typ/Type: `int`

- `x % y / 2`

  Wert/Value: `0`

- `x-- * ++y`

  Typ/Type: `int`

- `x-- * ++y`

  Wert/Value: `45`

Für die Fragen rechts, verwenden Sie den IEEE-Standard für Fliesskommazahlen.

For the questions on the right, assume the IEEE standard for floating point arithmetics.

false Das Literal `1e7f` bezeichnet denselben Wert wie das Literal `8e4f`.
The literal `1e7f` has the same value as the literal `8e4f`.

true Für zwei Fliesskommavariablen `a` und `b` mit beliebigem Wert gilt:
For any two floating point variables `a` and `b`, we have:

```
a * b == b * a
```

false Für drei Fliesskommavariablen `a`, `b`, und `c` mit beliebigem Wert gilt:
For any three floating point variables `a`, `b`, and `c`, we have:

```
a + b + c == c + b + a
```

false Für eine Fliesskommavariable `a` mit beliebigem Wert gilt:
For any floating point variable `a`, we have:

```
a + 1/2 == a + 0.5
```

false Der Typ `double` hat genau zwei mal so viele signifikante Bits wie der Typ `float`.
The type `double` has exactly twice as many significant bits as `float`.

false Für `a>0` vom Typ `double` ist `std::sqrt(a)` die exakte Wurzel von `a`.
For `a>0` of type `double`, `std::sqrt(a)` is the exact square root of `a`.

Betrachten Sie folgendes Programm. Beantworten Sie die Fragen auf der rechten Seite!
Hinweis: die Funktionen `proc1` und `proc2` unterscheiden sich im zweiten Argument.

```
#include <iostream>

void proc1(int* n, int &m)
{
  *n = *n + 5;
  std::cout << *n * m;
}

void proc2(int* n, int m)
{
  *n = *n + 5;
  std::cout << *n * m;
}

int fun(int x, int n)
{
  if (n == 1)
    return x;
  else
  {
    int y = fun(x, n / 2);
    return y * y;
  }
}

int main()
{
    ...
}
```

Consider the program above. Answer the questions on the right!
Note that the functions `proc1` and `proc2` differ in their second argument.

- Was gibt das Programm mit folgender `main`-Funktion aus?
  What does the program with the following `main` function print?

```
int main()
{
  int k = 5;
  proc1(&k, k);
  return 0;
}
```

100

- Was gibt das Programm mit folgender `main`-Funktion aus?
  What does the program with the following `main` function print?

```
int main()
{
  int k = 5;
  proc2(&k, k);
  return 0;
}
```

50

- Was gibt das Programm mit folgender `main`-Funktion aus?
  What does the program with the following `main` function print?

```
int main()
{
  std::cout << fun(2, 12);
  return 0;
}
```

256

Die folgende BNF definiert eine einfache Programmiersprache zur effizienten Beschreibung von Turtle-Grafiken. Die drei Schildkröten-Befehle $+$, $-$ und F können mit Hilfe geschweifter Klammern gruppiert und mit Hilfe natürlicher Zahlen wiederholt werden. Das Programm $\{3\{\mathrm{F}+\}\mathrm{F}\}$ beschreibt zum Beispiel die Befehlsfolge $\mathrm{F}+\mathrm{F}+\mathrm{F}+\mathrm{F}$. Beantworten Sie die Fragen auf der rechten Seite!

```
program = "{" sequence "}".
sequence = statement | statement sequence.
statement = repetitions body | body.
repetitions = unsigned int.
body = command | program.
command = "+" | "-" | "F".
```

The BNF above defines a simple programming language for the efficient description of turtle graphics. The three turtle commands $+$, $-$, and F can be grouped through curly braces, and repeated through natural numbers. For example, the program $\{3\{\mathrm{F}+\}\mathrm{F}\}$ describes the command sequence $\mathrm{F}+\mathrm{F}+\mathrm{F}+\mathrm{F}$. Answer the questions on the right side!

true Das folgende Programm ist gültig nach der BNF.
The following program is valid according to the BNF.
`{4F+}`

true Das folgende Programm ist gültig nach der BNF.
The following program is valid according to the BNF.
`{F-F-}`

true Das folgende Programm ist gültig nach der BNF.
The following program is valid according to the BNF.
`{{{2F}{2+}}}`

false Das folgende Programm ist gültig nach der BNF.
The following program is valid according to the BNF.
`{4{}}`

- Gib ein kürzestes Programm an, das nach der BNF gültig ist.
  Provide a shortest program that is valid according to the BNF.

  `{F}`

Wir nehmen an, dass folgende Funktionen eine Eingabe auf Gültigkeit gemäss der BNF der vorigen Aufgabe prüfen. Das gelte auch für die Funktionen ohne Definition. Beantworten Sie die Fragen auf der rechten Seite!

```
// POST: leading whitespace characters are extracted
//       from ‘is’, and the first non-whitespace character
//       is returned (0, if there is no such character)
//       the character is not consumed from the stream
char lookahead (std::istream& is);
// program = "{" sequence "}"
bool program(std::istream& s);
// sequence = statement | statement sequence
bool sequence(std::istream& s) {
  if (!statement(s)) return false;
  if (lookahead(s) != '}')
    return      A      ;
  return true;
}
// statement = repetitions body | body
// repetitions = unsigned int
bool statement(std::istream& s);
// body = command | program
bool body(std::istream& s) {
  char c = lookahead(s);
  if (      B      )
    return program(s);
  else
    return command(s);
}
// command = "+" | "-" | "F"
bool command(std::istream& s) {
  char c = lookahead(s);
  if (c == '+' ||      C      )
    return s >> c;
  return false;
}
```

We assume that the functions displayed above verify if an input is valid according to the BNF from the previous task. We assume this applies also to the functions without definition. Answer the questions on the right side!

- Welcher Ausdruck muss bei A eingesetzt werden?
  Fill in the expression for A.

```
sequence(s)
```

- Welcher Ausdruck muss bei B eingesetzt werden?
  Fill in the expression for B.

```
c == '{'
```

- Welcher Ausdruck muss bei C eingesetzt werden?
  Fill in the expression for C.

```
c == '-' || c == 'F'
```

Das Skalarprodukt zweier Vektoren $x, y \in \mathbb{R}^n$ ist definiert als

$$\langle x, y \rangle := \sum_{i=1}^{n} x_i y_i.$$

Gegeben ist das Gerüst eines Programms, das das Skalarprodukt zweier Vektoren berechnet. Beantworten Sie die Fragen auf der rechten Seite so, dass sich ein korrektes Programm ergibt!

```
// PRE:  [x_begin, x_end), [y_begin, y_end) are valid ranges
//       of the same size
// POST: the scalar product of the two ranges is returned
double scalar_product(const double* x_begin,
                      const double* x_end,
                      const double* y_begin,
                      const double* y_end) {
  double result = 0.0;
  const double* y = y_begin;
  for (const double* x = x_begin;  A  ; ++x) {
    B
    ++y;
  }
  return result;
}

int main() {
  double x[] = {1, 2, 3, 4, 5};
  double y[] = {6, 7, 8, 9, 10};
  std::cout <<  C  << "\n"; // 130
  return 0;
}
```

The scalar product of two vectors $x, y \in \mathbb{R}^n$ is defined as

$$\langle x, y \rangle := \sum_{i=1}^{n} x_i y_i.$$

The skeleton of a program for computing the scalar product of two vectors is provided above. Answer the questions on the right side such that you get a correct program!

- Welcher Ausdruck muss bei A eingesetzt werden?
  Fill in the expression for A.

```
x < x_end
```

- Welche Anweisung muss bei B eingesetzt werden?
  Fill in the statement for B.

```
result += *x * *y;
```

- Welcher Ausdruck muss bei C eingesetzt werden, damit der Wert des Skalarproduktes ausgegeben wird?
  Fill in the expression for C such that the value of the scalar product is output.

```
scalar_product(x,x+5,y,y+5)
```

Ein nichtleerer Stapel erlaubt die pop-Operation (Entfernung des obersten Elements). Ein Stapel mit mindestens zwei Elementen erlaubt auch die Operation rob (Entfernung des zweitobersten Elements). In dieser Aufgabe geht es darum, die rob-Operation zu implementieren. Vorgegeben sind Gerüste der Klassen linked_list_node und stack mit den relevanten Member-Variablen und der Member-Funktion rob. Lesen und beantworten Sie die Fragen auf der rechten Seite so, dass sich eine korrekte Implementierung von rob ergibt!

```
struct linked_list_node {
  int key;
  linked_list_node* next;
  ...
};

class stack {
public:
    ...
    // PRE:  *this has at least two elements
    // POST: the element below the top element is deleted
    //       and removed from *this
    void rob()
    {
      assert(      A      );
      linked_list_node* p =      B      ;
           C
      delete p;
    }
private:
    linked_list_node* top_node;
};
```

A non-empty stack allows the pop operation (removal of the top element). A stack with at least two elements also allows the rob operation (removal of the element below the top). This task is about implementing the rob operation. You are given skeletons of the classes linked_list_node and stack, with the relevant member variables and the member function rob. Read and answer the questions on the right side such that you get a correct implementation of rob!

- Welcher Ausdruck muss bei A eingesetzt werden? Die zugehörige assert Anweisung soll prüfen, dass mindestens zwei Elemente auf dem Stack liegen.
  Fill in the expression for A. The corresponding assert statement shall check that at least two elements are present on the stack.

```
top_node != 0 && top_node->next != 0
```

- Welcher Ausdruck muss bei B eingesetzt werden?
  Fill in the expression for B.

```
top_node -> next
```

- Welche Anweisung muss bei C eingesetzt werden?
  Fill in the statement for C.

```
top_node->next = p->next;
```

IBANs (International Bank Account Numbers) sind Zeichenketten aus Ziffern und Grossbuchstaben, z.B. "CH3900700115651849173". Zur Validierung wird eine IBAN in eine Ganzzahl umgewandelt. Grossbuchstaben A..Z werden durch die Zahlen 10...35 ersetzt. "789XYZ" wird z.B. zu 789333435. Betrachten Sie folgenden Code und beantworten Sie die Fragen auf der rechten Seite.

```
// PRE: c is either a digit 0...9 or a capital letter A...Z
// POST: if c is a digit, the digit value is returned;
// otherwise A yields 10, ..., Z yields 35
int IBAN_integer(char c) {
    if (std::isdigit(c))
        return      A      ;
    else
        return      B      ;
}

// PRE: s consists of digits 0...9 and capital letters A...Z
// POST: the IBAN integer of s is returned, obtained by
// replacing any A with 10, ..., any Z with 35
int IBAN_integer(const std::string &s) {
    int result = 0;
    for (int i = 0; i < s.length(); ++i) {
        int next = IBAN_integer(s[i]);
        if (next < 10)
            result =      C      ;
        else
            result =      D      ;
    }
    return result;
}
```

IBANs (International Bank Account Numbers) are strings of digits and capital letters, e.g., "CH3900700115651849173". For validation, an IBAN is converted to an integer, by replacing capital letters A...Z with integers 10...35. For example, the substring "789XYZ" yields the integer 789333435. Consider the code above and answer the questions on the right hand side.

- Ergänzen Sie die fehlenden Ausdrücke, so dass die richtige **Konversion von Zeichen zu Ganzzahl** ausgeführt wird.

  Annahme: Zeichen sind mit ASCII codiert: Ziffern 0...9 haben ASCII codes 48...57 und Buchstaben A...Z haben ASCII Codes 65...90.

  Provide the missing expressions such that the correct **conversion from character to integer** is executed.

  Assumptions: Characters are encoded in ASCII: digits 0...9 have ASCII codes 48...57, and capital letters A...Z have ASCII codes 65...90.

  Ausdruck für **A**: `c - 48`

  Ausdruck für **B**: `c-65+10`

- Ergänzen Sie die fehlenden Ausdrücke, so dass die richtige **Konversion von Zeichenkette zu Ganzzahl** ausgeführt wird.

  Annahmen: die resultierenden Ganzzahlen passen immer in eine Variable vom Typ 'int'.

  Provide the missing expressions such that the correct **conversion form string to integer** is executed.

  Assumptions: resulting integers always fit into a variable of type 'int'.

  Ausdruck für **C**: `result * 10 + next`

  Ausdruck für **D**: `result * 100 + next`

Bei der Validierung einer IBAN wird eine Modulo-Operation auf eine grosse Zahl angewendet. Die Zahl ist üblicherweise zu gross, um in einem `int` gespeichert zu werden. Wir nehmen nun an, die Ziffern der Zahl seien in einem Ganzzahlfeld gespeichert. Die Funktion `IBAN_mod97` sollte die entsprechende Zahl modulo 97 berechnen.

Ein Beispiel ist in der `main` Funktion zu sehen.

Beantworten Sie die Fragen auf der rechten Seite.

```
#include <iostream>

// PRE: [begin, end) is a valid range of
//      an array of integer values between 0 and 9
// POST: returns the number formed by the digits
//      in [begin,end) modulo 97
int IBAN_mod97(int* begin, int* end)
{
    int result = 0;
    while (begin < end)
    {
        A
        ++begin;
    }
    return result;
}

int main()
{
    int a[] = {1,2,3,8,3,2,1,9,8,8,6,2};
    // computation of 123832198862 mod 97 = 80 :
    std::cout << IBAN_mod97(a,a+12); // 80
}
```

For the validation of an IBAN number a modulo operation is applied to a large number. The number is usually too large to be stored in a single `int`. We assume that the digits of the number are stored in an integer array. The function `IBAN_mod97` should return the corresponding number modulo 97.

An example is given in the `main` function above.

Answer the questions on the right hand side.

- Welche Anweisung muss bei A eingesetzt werden?
  Fill in the statement for A.

```
result = (result * 10 + *begin) % 97;
```

Hinweis: Für ganze Zahlen a, b und $n > 0$ gelten folgende Regeln:
Hint: for integers a, b and $n > 0$ the following rules apply:

$$(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$$
$$(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$$

- Es gilt $12345678900 \bmod 97 = 20$. Durch welche beiden Zahlen müssen die letzten beiden Nullen in

  `int b[] = {1,2,3,4,5,6,7,8,9,0,0};`

  ersetzt werden, so dass `IBAN_mod97(b,b+11)` den Wert 1 zurückgibt?

  It holds that $12345678900 \bmod 97 = 20$. By which two numbers do the two last zeros in

  `int b[] = {1,2,3,4,5,6,7,8,9,0,0};`

  have to be replaced such that `IBAN_mod97(b,b+11)` returns value 1?

```
78
```