立志舎
「SUKI-1 GRAND PRIX」
プログラミングバトル
出題範囲

【Python3】

プログラミングの知識

標準出力

```Python
print("Hello world!") # 文字列の出力
print(12345) # 整数の出力
```

四則演算とあまりの計算

```Python
print(1 + 5)   # 足し算
print(7 - 2)   # 引き算
print(5 * 3)   # 掛け算
print(6 // 3)  # 整数の割り算(小数点以下は切り捨て)
print(7 % 2)   # あまり
```

変数と型

```Python
a = 10       # 整数を扱う変数の定義
b = a * 3    # 変数を用いた計算
c = "aiueo" # 文字列を扱う変数の定義
a = 13456789 * 987654321 # 巨大な数(32bitを超える数)の扱い方。64bit以上の計算は今回出題されません。
```

標準入力

Python

```python
# 1行に半角空白区切りで整数がある場合（各変数に格納）
a, b, c = map(int, input().split())

# 整数を受け取る
d = int(input())

# 文字列を受け取る
e = input()


# 1行に半角空白区切りで整数がN個ある場合
A = list(map(int, input().split()))

# N × Nの2次元配列を受け取る
B = [list(map(int, input().split())) for _ in range(N)]


# クエリ（問い合わせ）を受け取る
Q = int(input())
query = [input().split() for _ in range(Q)]
```

条件分岐と比較演算子

Python

```python
if(a < b):
	# if文
elif(a > b):
	# elif文
else:
	# else文

# 比較演算子
if(a == b): # aとbが等しい場合
if(a != b): # aとbが異なる場合
if(a < b):  # aがb未満の場合
if(a <= b): # aがb以下の場合
if(a >= b): # aがb以上の場合
if(a > b):  # aがbより大きい場合
```

ループ

Python

```
for i in range(n):
       # for文(n回の繰り返し)

while(a > 0):
  # while文(条件を満たす間の繰り返し)
```

配列(リスト)

Python

```
a = [0, 1, 2, 3, 4, 5] # 配列(リスト)の定義
b = [0 for i in range(n)] # N要素の配列の定義
c = map(int, input().split()) # 整数配列を標準入力から受け取り
d = [[0 for i in range(m)] for i in range(n)] # 多次元配列の定義

x = a[3]    # 配列の各要素へのアクセス
y = d[3][4] # 多次元配列の要素のアクセス
a[4] = 10   # 配列の要素の書き換え

a[1:4] # [1, 2, 3]
a[2:]  # [2, 3, 4, 5]
a[:3]  # [0, 1, 2]
a[-1]  # [5]

a.append(10) # 配列aの末尾に数字を追加する（この場合、10を追加）
a.pop() # 配列aの末尾から1つ削除
```

文字列

Python

```
s = "aiueo"
c = s[3]   # 各文字のアクセス
s += "z"   # 文字列の末尾に文字(文字列)の追加
number = int("123") # 数字の文字列をintで囲うと数値型に変換できる

l = len("abcde")    # len()を使うと、文字列の長さを取得できる
```

関数・メソッド

Python

```
a = [2, 3, 1, 4, 5]
```

```
s = sum(a) # sum()を使うと配列の合計を取得することができる（この場合15となる）
m = min(a) # min()を使うと配列の最小値を取得することができる（この場合1となる）

s1 = sorted(a) # sorted()を使うと配列の昇順に並び替えることができる
a.reverse() # reverse()を使うと配列の逆順に並び替えることができる
```

数学の知識

- 一次方程式や一次不等式の計算方法
- 倍数と約数の意味

アルゴリズムに関する知識(※優勝を目指す方へ向けて)

- 配列の要素の並び替え方法(ソート)
- プログラムの効率の見積もりと改善(5秒以内に実行完了するアルゴリズムの考案)
    - 特定のアルゴリズムに関する知識を要する問題は今回出題されません。発想力があれば解答できる問題のみが、出題されます。

Python
```
# 例：1~Nの和を計算する方法

# プログラム1
a = 0
for i in range(1, n+1):
 a += i
# ↑この方法だとn回の計算が必要なので、nが大きくなるにつれ時間がかかる。

# プログラム2
a = (n*(n+1))//2
# ↑この方法であれば1回の計算のみなのでnがどれだけ大きくなっても時間がかからない。
```

※その他のプログラミングの構文や知識があれば、有利に解ける問題が出題されることがあります。

【C】

プログラミングの知識

標準出力

C/C++
```c
#include <stdio.h>

int main() {
    printf("Hello world!\n");  // 文字列の出力
    printf("%d\n", 12345);  // 整数の出力
    return 0;
}
```

四則演算とあまりの計算

C/C++
```c
printf("%d\n", 1 + 5);  // 足し算
printf("%d\n", 7 - 2);  // 引き算
printf("%d\n", 5 * 3);  // 掛け算
printf("%d\n", 6 / 3);  // 整数の割り算(小数点以下は切り捨て)
printf("%d\n", 7 % 2);  // あまり
```

変数と型

C/C++
```c
int a = 10;           // 整数を扱う変数の定義
int b = a * 3;        // 変数を用いた計算
char c[6] = "aiueo";  // 文字列を扱う変数の定義
long long d = 12345678911 * 98765432111;  // 巨大な数(32bitを超える数)の扱い方。
64bit以上の計算は今回出題されません。
```

標準入力

C/C++
```c
/* rewindは使用できません。ご注意ください。 */

// 整数1つ受け取る
int x;
scanf("%d", &x);
```

```c
// 半角空白区切りの整数を受け取る
int a, b, c;
scanf("%d %d %d", &a, &b, &c);


// 長さNの文字列を受け取る
char S[N + 1];
scanf("%s", S);

// N個の整数を受け取る
int A[N];

for (int i = 0; i < N; i++) {
    scanf("%d", &A[i]);
}

// N × N の二次元配列
int A[N][N];

for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        scanf("%d", &A[i][j]);
    }
}

// Q個のクエリ（問い合わせ）を受け取る
int Q;
scanf("%d", &Q);

for(int i = 0; i < Q; i++) {
    int c, x;
    scanf("%d", &c);

    if(c == ?) {    // cが該当する値ならxも受け取る
        scanf("%d", &x);
    }
}
```

条件分岐と比較演算子

C/C++

```c
if(a < b){
    // if文
}else if(a > b){
    // elif文
```

```
}else{
    // else文
}

if(a == b)  // aとbが等しい場合
if(a != b)  // aとbが異なる場合
if(a < b)   // aがb未満の場合
if(a <= b)  // aがb以下の場合
if(a >= b)  // aがb以上の場合
if(a > b)   // aがbより大きい場合
```

ループ

```
C/C++
for(i = 0; i <= n; i++){
    // for文(n回の繰り返し)
}

while(a > 0){
    // while文(条件を満たす間の繰り返し)
}
```

配列(リスト)

```
C/C++
int a[6] = {1, 2, 3, 4, 5, 6};  // 配列(リスト)の定義
int b[100] = {};  // 0で初期化した配列の定義

int c[100] = {};
for(i = 0; i < n; i++){
    scanf("%d", &c[i]);  // 整数配列を標準入力から受け取り
}

int d[100][100] = {};  // 多次元配列の定義

x = a[3];  // 配列の各要素へのアクセス
y = d[3][4];  // 多次元配列の要素のアクセス
a[4] = 10;  // 配列の要素の書き換え
```

文字列

```
C/C++
char s[10] = "aiueo";
```

```
char c = s[3];  // 各文字のアクセス

int i = strlen(s);  // 文字列の長さ取得
s[i] = 'z';  // 文字列の末尾に文字の追加

int n = s[0] - 'a'  // 文字を数値に変換できる（aの場合0、bの場合1...zの場合25となる）
```

関数・メソッド

```
C/C++
#include <string.h>

// strlenで文字列の長さ取得できる（#include <string.h>が必要！）
int l = strlen("abcd")
```

数学の知識

- 一次方程式や一次不等式の計算方法
- 倍数と約数の意味

アルゴリズムに関する知識(※優勝を目指す方へ向けて)

- 配列の要素の並び替え方法(ソート)
- プログラムの効率の見積もりと改善(5秒以内に実行完了するアルゴリズムの考案)
    - 特定のアルゴリズムに関する知識を要する問題は今回出題されません。発想力があれば解答できる問題のみが、出題されます。

```
C/C++
// 例：1~Nの和を計算する方法

// プログラム1
int a = 0;
for(i = 1; i <= n; i++){
    a += i;
}
// ↑この方法だとn回の計算が必要なので、nが大きくなるにつれ時間がかかる。

// プログラム2
int b = (n*(n+1))/2;
```

```
// ↑この方法であれば1回の計算のみなのでnがどれだけ大きくなっても時間がかからない。
```

※その他のプログラミングの構文や知識があれば、有利に解ける問題が出題されることがあります。

【Java】

プログラミングの知識

標準出力

```Java
public class Main {
    public static void main(String args[] ) throws Exception {
        System.out.println("Hello world!");  // 文字列の出力
        System.out.println(12345);  // 整数の出力
    }
}
```

四則演算とあまりの計算

```Java
System.out.println(1 + 5);  // 足し算
System.out.println(7 - 2);  // 引き算
System.out.println(5 * 3);  // 掛け算
System.out.println(6 / 3);  // 整数の割り算(小数点以下は切り捨て)
System.out.println(7 % 2);  // あまり
```

変数と型

```Java
int a = 10;  // 整数を扱う変数の定義
int b = a * 3;  // 変数を用いた計算
String c = "aiueo";  // 文字列を扱う変数の定義
long d = 123456789L * 987654321L;  // 巨大な数(32bitを超える数)の扱い方。64bit以上の計算は今回出題されません。
```

標準入力

```Java
Scanner sc = new Scanner(System.in);

// 整数を1つ受け取る
int a = sc.nextInt();

// 文字列を1つ受け取る
String s = sc.next();
```

```java
// N個の整数を受け取る
int[] A = new int[N];

for (int i = 0; i < N; i++) {
    A[i] = sc.nextInt();
}


//  N × N の2次元配列を受け取る
int[][] A = new int[N][N];

for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        A[i][j] = sc.nextInt();
    }
}

// Q個のクエリ（問い合わせ）を受け取る
int Q = sc.nextInt();

for(int i = 0; i < Q; i++) {
    int c = sc.nextInt();  // cを受け取る

    if(c == ?) {    // cが該当の数値ならxも受け取る
        int x = sc.nextInt();
    }
}
```

条件分岐と比較演算子

Java

```java
// 条件分岐と比較演算子
if(a < b){
    // if文
}else if(a > b){
    // elif文
}else{
    // else文
}

if(a == b)  // aとbが等しい場合
if(a != b)  // aとbが異なる場合
if(a < b)   // aがb未満の場合
if(a <= b)  // aがb以下の場合
```

```
if(a >= b)  // aがb以上の場合
if(a > b)   // aがbより大きい場合
```

ループ

Java

```
for(i = 0; i < n; i++){
    // for文(n回の繰り返し)
}

while(a > 0){
    // while文(条件を満たす間の繰り返し)
}
```

配列(リスト)

Java

```
int[] a = {1, 2, 3, 4, 5, 6}; // 配列の定義
int[] b = new int[100]; // 0で初期化した配列の定義

int[] c = new int[100];
for (int i = 0; i < n; i++) {
    c[i] = sc.nextInt();  // 整数配列を標準入力から受け取り
}

int[][] d = new int[100][100]; // 多次元配列の定義

x = arrA[3];  // 配列の各要素へのアクセス
y = arrD[3][4];  // 多次元配列の要素のアクセス
a[4] = 10;  // 配列の要素の書き換え


List<Integer> l = new ArrayList<>();    // リストの宣言
l.add(1)     // リストの末尾に追加
Collections.sort(l) // リストを昇順にソート（並び替え）する
l.remove(l.size() - 1)     // リストの末尾を削除する
```

文字列

Java

```
String s = "aiueo";
char c = s.charAt(3);  // 各文字のアクセス
s = s + "z"; // 文字列の末尾に文字(文字列)の追加
```

```
s.charAt(0) - 'a'    // 文字を数値に変換できる（aの場合0、bの場合1...zの場合25となる）

int s = s.length(); // length()を用いることで文字列の長さを取得できる
```

関数・メソッド

```Java
int a = Math.min(3, 2); // Math.minを使うと最小値を取得できる（この場合、2が取得できる）
```

数学の知識

- 一次方程式や一次不等式の計算方法
- 倍数と約数の意味

アルゴリズムに関する知識(※優勝を目指す方へ向けて)

- 配列の要素の並び替え方法(ソート)
- プログラムの効率の見積もりと改善(5秒以内に実行完了するアルゴリズムの考案)
    - 特定のアルゴリズムに関する知識を要する問題は今回出題されません。発想力があれば解答できる問題のみが、出題されます。

```Java
// 例: 1~Nの和を計算する方法

// プログラム1
int a = 0;
for(i = 1; i <= n; i++){
    a += i;  // 1からnまでの整数を順に加算
}
// ↑この方法だとn回の計算が必要なので、nが大きくなるにつれ時間がかかる。

// プログラム2
int b = (n*(n+1))/2;
// ↑この方法であれば1回の計算のみなのでnがどれだけ大きくなっても時間がかからない。
```

※その他のプログラミングの構文や知識があれば、有利に解ける問題が出題されることがあります。