2020 年度 夏入試 / 2020 Summer Entrance Examination

東京大学情報理工学系研究科創造情報学専攻
Department of Creative Informatics
Graduate School of Information Science and Technology, The University of Tokyo

# プログラミング / Programming

## 注意事項 / INSTRUCTIONS

1. 試験開始の合図まで，この問題冊子を開いてはいけない．
   Do not open this problem booklet until the start of the examination is announced.
2. 英文ページは和文ページの後である．
   The English pages follow the Japanese pages.
3. この表紙の下部にある受験番号欄，解答用紙および草稿用紙に受験番号を記入すること．
   Write your examinee ID number below on this cover page, an answer sheet and a draft sheet.
4. プログラミング言語は何を使ってもよい．
   You may choose any programming language to answer.
5. プログラミングの本は 1 冊に限り試験中に参照してもよい．インターネットに PC を接続してはいけないが，各自の PC に入っているライブラリやプログラム断片を使用・流用することは自由である．
   During the examination, you may consult only one printed book on programming. You can use or copy any libraries or program fragments stored in your PC but **you may not connect the PC to the Internet.**
6. 試験終了時までに，自分の PC 上に受験番号名のディレクトリ/フォルダを作成し，作成したプログラムおよび関連ファイルをその下に保存すること．作成したディレクトリ/フォルダを渡された USB メモリにコピーすること．
   By the end of the examination, make a directory/folder on your PC. Use your examinee ID number as its name. Store your program files and related files into the directory/folder. **Copy** the directory/folder **onto the USB flash drive** that you received.
7. 部屋から退出するときは，PC と問題冊子を残すこと．
   **Leave your PC and this booklet** in the room on the exit.

受験番号/Examinee ID number _____

このページは空白.

This page is blank.

このページは空白.

This page is blank.

# プログラミング

以下の問いに必要ならプログラムを書いて答えよ。書いたプログラムを試験終了前に USB メモリに保存すること。

(1) バイナリ・データをテキスト・ファイルとして保存する。バイナリ・データを 6 bits ずつ区切り、各 6 bits 000000 から 111111 を順に文字 A, B, ... Z, a, b, ... z, 0, 1, ... 9, @, # に置換してファイルに保存するとする。例えば 6 bits が 000000 なら A, 000001 なら B, 000010 なら C, 000111 なら H, 111110 なら @, 111111 なら # に置換する。またバイナリ・データが

000001000111000010111111

というビット列ならば、テキスト・ファイルには BHC# が保存される。バイナリ・データの bit 長は 6 の倍数とする。

あるバイナリ・データが上記の形式でテキスト・ファイル data1.txt に保存されている。元のバイナリ・データの 310 bit 目から 320 bit 目までのビット列 (11 bits) を解答用紙に書け。バイナリ・データの左端先頭を 0 bit 目とする。

(2) バイナリ・データを圧縮してファイルに保存する。圧縮されたファイルを復元するプログラムは、圧縮されたファイルの先頭から 1 byte (8 bits) ずつ読み、復元されたファイルの末尾に次のようにデータを追加する。

- 読んだ 1 byte が 0 以外のときは、その 1 byte をそのまま追加する。
- その 1 byte が 0 のとき、続く 2 bytes を 8 bits の符号なし整数 2 つとしてファイルから読み、その値を $p$ と $d$ とする。ただし常に $256 > p \geq d \geq 0$ とする。
  - $d = 0$ のとき $p$ の値にかかわらず 1 byte のバイナリ・データ 0 を追加する。
  - それ以外の場合、それまでに復元されたファイルの末尾から $p$ bytes 前から $p - d + 1$ bytes 前までの部分 byte 列の複製を、末尾に追記する。復元されたファイルの末尾に最後に追加された byte を 1 byte 前 ($p = 1$) とする。

例えば圧縮されたバイナリ・ファイルの中身の各 byte を 16 進数で表すと

41 42 43 44 45 46 47 00 06 05 48

であるとき、復元されたファイルの中身は

41 42 43 44 45 46 47 42 43 44 45 46 48

である。

圧縮されたバイナリ・ファイルを上記の方法で復元し、復元後のファイルの大きさ (byte) を表示するプログラムを書け。また圧縮されたバイナリ・ファイル data2a.bin、data2b.bin、data2c.bin をそのプログラムで復元し、復元後のファイルのそれぞれの大きさ (byte) を解答用紙に書け。復元後のファイルはそれぞれ、data2a.txt、data2b.tif、data2c.txt とし、USB メモリに保存せよ。

(3) 与えられたバイナリ・ファイルを圧縮し、圧縮後のファイルの大きさ (byte) を表示するプログラムを書け。圧縮されたファイルは (2) で書いたプログラムで復元される。プログラムはファイルをなるべく小さくなるように圧縮する。バイナリ・ファイル data3a.txt、data3b.png、data3c.txt をそのプログラムで圧縮し、圧縮後のファイルのそれぞれの大きさ (byte) を解答用紙に書け。また圧縮後のファイルはそれぞれ data3a.bin、data3b.bin、data3c.bin とし、USB メモリに保存せよ。

(4) 英語の文章を単一換字式暗号で暗号化してファイルに保存する。文章は小文字 a から z と、ピリオド . そして空白からなる。文の最後はピリオドで終わる。単一換字式暗号では、各文字を別の決まった文字（アルファベット小文字またはピリオド、空白のいずれか）に置き換えて暗号化する。置換後の文字は同じ文字のこともある。

data4.txt はそのような方法で暗号化した英文のテキスト・ファイルである。data4.txt を復号して得られる平文に出現する全ての単語の一覧 data4dict.txt (空白区切り) を参考に data4.txt の英文を復号し、得られた平文の第一文を解答用紙に書け。

(5) バイナリ・ファイルを暗号化する。バイナリ・ファイルの中身を 4 bytes ずつ区切り、それぞれを次のように暗号化する。バイナリ・ファイルの大きさ (byte) は 4 の倍数とする。まず 4 bytes の各 byte を 8 bit の符号なし整数とみなし、先頭から $b_0, b_1, b_2, b_3$ とする。

$$m = \sum_{k=0}^{3} 2^{8(3-k)} b_k$$

とし、$e = 551263368336670859257571$, $n = 385884357836063206955733 7$ として、

$$c = m^e \ mod \ n$$

とする。$n$ は秘密の素数 $p$ と $q$ の積である。また $mod\ n$ は $n$ を法とする剰余である。なお

$$(x \times y) \ mod \ n = ((x \ mod \ n) \times (y \ mod \ n)) \ mod \ n$$

である。

暗号化されたファイルは、区切られた各 4 bytes から計算される $c$ を 10 進数の文字列として順に並べたテキスト・ファイルとする。各 $c$ を表す文字列は空白 1 文字で区切られる。例えば元のバイナリ・ファイルの中身の各 byte を 16 進数で表すと

41 42 43 44 45 46 47 48

であるとき、暗号後のファイルはテキスト・ファイルで中身は

367829405937736238906682 7 320604555002205363990110 8

である。

復号には秘密の整数 $d$ を用いる。この $d$ について

$$m = c^d \ mod \ n$$

が成り立つ。この暗号は秘密の整数が推測されると破られる。今、$e \times d = (p-1)(q-1)+1$ であることがわかっている。この事実を利用して data5.txt を復号せよ。復号後のデータは UTF-8 テキストである。このテキストを解答用紙に書け。

# Programming

Answer the following questions by writing programs if necessary. Store the programs in the USB flash drive before the examination ends.

(1) We store binary data in a text file. We split binary data to 6-bit chunks and store them in the file after replacing every 6-bit number, `000000` to `111111`, with a character `A`, `B`, ... `Z`, `a`, `b`, ... `z`, `0`, `1`, ... `9`, `@`, or `#`, respectively, in ascending order. For example, we replace the 6-bit number with `A` when it is `000000`, `B` when it is `000001`, `C` when it is `000010`, `H` when it is `000111`, `@` when it is `111110`, and `#` when it is `111111`. When the binary data is a bit sequence:

```
000001000111000010111111
```

we store `BHC#` in the text file. The bit length of the binary data is a multiple of 6.

The text file `data1.txt` stores binary data in the format shown above. Obtain the bit sequence (11 bits) from the 310th bit to the 320th bit of that binary data, and write the sequence on the answer sheet. The left-most bit of the binary data is the 0th bit.

(2) We store binary data in a file after compressing them. The program restoring the compressed file reads every byte (8 bits) from the beginning of the compressed file, and appends data to the end of the restored file as follows:

- Append the read byte as it is unless the byte is 0,
- When the byte is 0, read the following two bytes as two 8-bit unsigned integers from the file. Let them $p$ and $d$. It always holds $256 > p \geq d \geq 0$.
  - When $d = 0$, append 1-byte binary data 0 no matter what the value of $p$ is.
  - Otherwise, append a copy of the sub-sequence of bytes from the $p$-th byte to the $(p - d + 1)$-th byte counting from the end of the file already restored so far. The byte last appended to the restored file is the first byte ($p = 1$).

For example, when the bytes stored in the compressed binary file are:

```
41 42 43 44 45 46 47 00 06 05 48
```

in the hexadecimal form, the restored file stores the following bytes:

```
41 42 43 44 45 46 47 42 43 44 45 46 48
```

Write the program that restores a compressed binary file by the method shown above, and prints the size (bytes) of the file after the restoration. Restore the compressed binary files `data2a.bin`, `data2b.bin`, and `data2c.bin` by that program, and write their sizes (bytes) after the restoration down on the answer sheet. After the restoration, name the files `data2a.txt`, `data2b.tif`, and `data2c.txt`, respectively. Store them in the USB flash drive.

(3) Write the program that compresses the given binary file and prints the size (bytes) of the file after the compression. The compressed file is restored by the program written for (2). The program compresses the file to be as small as possible. Compress the binary file `data3a.txt`, `data3b.png`, and `data3c.txt` by that program, and write their sizes after the compression down on the answer sheet. After the compression, name the files `data3a.bin`, `data3b.bin`, `data3c.bin`, respectively. Store them in the USB flash drive.

(4) We encrypt English text by a simple substitution cipher and store the encrypted text in a file. The text consists of lower-case letters a to z, periods ., and/or white space characters. A sentence ends with a period. A simple substitution cipher encrypts the text by replacing each letter with another fixed letter (lower-case alphabets, a period, or a white space character), which may be the same letter.

The text file `data4.txt` stores a cipher text encrypted by this method. Decrypt `data4.txt` by referring to `data4dict.txt` (white-space separated), which lists all the words included in the plaintext obtained by decrypting `data4.txt`, and write the first sentence of the obtained plaintext down on the answer sheet.

(5) We encrypt a binary file. We split the contents of the binary file to 4-byte chunks and encrypt each chunk as follows. The size (bytes) of the binary file is a multiple of 4. First, read each byte of the four bytes as an 8-bit unsigned integer and let them $b_0, b_1, b_2$, and $b_3$, respectively from the beginning. Then, let:

$$m = \sum_{k=0}^{3} 2^{8(3-k)} b_k.$$

Let $e = 551263368336670859257571$, $n = 385884357836063206955733 7$, and

$$c = m^e \ mod \ n.$$

Here, $n$ is the product of secret prime numbers $p$ and $q$. $mod\ n$ expresses modulo $n$. Note that it holds

$$(x \times y) \ mod \ n = ((x \ mod \ n) \times (y \ mod \ n)) \ mod \ n.$$

The encrypted file is a text file storing the decimal numbers $c$ computed from the 4-byte chunks in the same order. The character strings expressing $c$ are separated by a white space character.

For example, when the original binary file stores the following bytes:

```
41 42 43 44 45 46 47 48
```

in the hexadecimal form, the encrypted file is a text file storing the following text:

```
367829405937736238906682 7 320604555002205363990110 8
```

The decryption uses a secret integer $d$. For this $d$, it holds:

$$m = c^d \ mod \ n.$$

This encryption is cracked if the secret integer is guessed. Now, we know that it holds $e \times d = (p-1)(q-1)+1$. Decrypt `data5.txt` by using this fact. The decrypted data is UTF-8 text. Write the text down on the answer sheet.

このページは空白.

This page is blank.

このページは空白.

This page is blank.