平 成 ３１ 年 度

名古屋大学大学院情報学研究科
知能システム学専攻
入学試験問題（専門）

平成３０年８月８日　50分

注意事項

１．試験開始の合図があるまでは、この問題冊子を開いてはならない。

２．試験終了まで退出できない。

３．外国人留学生は、語学辞書１冊に限り使用してよい。電子辞書の持ち込みは認めない。

４．外国人留学生は、英語での解答を可とする。

５．問題冊子、解答用紙３枚、草稿用紙３枚が配布されていることを確認すること。

６．問題は解析・線形代数、確率・統計、プログラミングの３科目がある。これらの全てについて解答すること。なお、解答した科目名を解答用紙の指定欄に記入すること。

７．全ての解答用紙の所定の欄に受験番号を必ず記入すること。解答用紙に受験者の氏名を記入してはならない。

８．解答用紙に書ききれない場合は、裏面を使用してもよい。ただし、裏面を使用した場合は、その旨、解答用紙表面右下に明記すること。

９．解答用紙は試験終了後に３枚とも提出すること。

１０．問題冊子、草稿用紙は試験終了後に持ち帰ること。

プログラミング

プログラムPは与えられた文字列を操作するC言語プログラムである．プログラムPで扱う文字は，1バイト，または，3バイトで構成される．1バイトで構成される文字は最上位ビットが0であり，残りのビットにより文字を指定する．3バイトで構成される文字は，その1バイト目の最上位ビットが1であり，1バイト目の残りのビットと2バイト目，ならびに，3バイト目により文字を指定する．文字列は8ビット長のchar型の配列に格納され，'¥0'を終端とする．16進数は，0x41のように先頭に0xをつけて表す．すなわち，0x41は10進表記の65の16進表記である．

プログラムPにおいて，27行目の関数concatは，ポインタs1，ポインタs2がこの順で引数として与えられたとき，s1とs2が参照する文字列をこの順に連結し，得られた文字列の先頭を参照するポインタを返す．59行目の関数replaceは，ポインタs1，ポインタs2，正の整数kがこの順で引数として与えられたとき，s1が参照する文字列のk文字目をs2が参照する文字列で置き換えて得られる文字列の先頭を参照するポインタを返す．なお，関数replaceの引数s2が参照する文字列の文字数は1とし，1文字は1バイトまたは3バイトで構成されることに注意せよ．

このとき以下の問いに答えよ．なお，ポインタが参照する文字列の内容を解答する場合，下の記述例のようにアドレスと値を対応づけて文字列を表すこと（終端も含めること）．例えば，80行目のr = str1;の実行直後のポインタrが参照する文字列の内容は，下の記述例のように解答する．

アドレス	r	r+1	r+2	r+3
値	0x41	0x42	0x43	'¥0'

(1) 空欄A，B，C，D，Eにあてはまる式を書いて，関数concatを完成させよ．

(2) 76行目のr = index(str1, 1);を実行した直後のrが参照する文字列の内容を記述例にならって書け．

(3) 77行目のr = index(str2, 2);を実行した直後のrが参照する文字列の内容を記述例にならって書け．

(4) 78行目のr = index(str3, 3);を実行した直後のrが参照する文字列の内容を記述例にならって書け．

(5) 79行目のr = replace(str3, str4, 3);の実行時に呼び出される62行目のtmp2 = concat_p(string1, string2, tmp1);を実行した直後のtmp2の内容を記述例にならって書け．

(6) 48行目では下線部のように固定長の領域をresultに割り当てている．しかしながら，関数concat_pに与えられる引数によっては領域が不足する．与えられる引数に応じて必要な領域が割り当てられるように下線部を適切な式に変更せよ．

(7) 空欄F，G，Hにあてはまる式を書いて関数replaceを完成させよ．

プログラムP

```
1   #include <stdlib.h>
2
3   char* index(char* string, int p){
4     char *tmp;
5     int i;
6     i = 1;
7     tmp = string;
8
9     while ( (*tmp != '¥0') && (i < p) ){
10      i++;
11      if ((*tmp & 0x80) == 0)
12        tmp++;
13      else
14        tmp += 3;
15    }
16    return tmp;
17  }
18
19  int length_b(char* string){
20    int len;
21    len = 0;
22    while (string[len] != '¥0')
23      len++;
24    return len;
25  }
26
27  char* concat(char* string1, char* string2){
28    char *result, *tmp;
29    result = (char*)malloc(sizeof(char) * (length_b(string1) + length_b(string2) + 1));
30    tmp = result;
31    while (*string1 != '¥0'){
32      [ A ] = [ B ];
33      tmp++;
34      string1++;
35    }
36    while (*string2 != '¥0'){
37      [ C ] = [ D ];
```

```
38      tmp++;
39      string2++;
40    }
41    *tmp = [      E      ];
42    return result;
43  }
44
45  char* concat_p(char* string1, char* string2, char* l){
46    char *result, *tmp1, *tmp2;
47    tmp1 = string1;
48    result = (char*)malloc(sizeof(char) *  100 );
49    tmp2 = result;
50    while ((tmp1 != l) && (*tmp1 != '¥0')){
51      *tmp2 = *tmp1;
52      tmp1++;
53      tmp2++;
54    }
55    *tmp2 = '¥0';
56    return concat(result, string2);
57  }
58
59  char* replace(char* string1, char* string2, int p){
60    char *tmp1, *tmp2;
61    tmp1 = index(string1, p);
62    tmp2 = concat_p(string1, string2, tmp1);
63    if ( [      F      ] == 0)
64      [      G      ];
65    else
66      [      H      ];
67    return concat(tmp2, tmp1);
68  }
69
70  void main(){
71    char str1[4] = {0x41, 0x42, 0x43, '¥0'};
72    char str2[10] = {0xE3, 0x81, 0x82, 0xE3, 0x81, 0x84, 0xE3, 0x81, 0x86, '¥0'};
73    char str3[9] = {0xE3, 0x81, 0x82, 0x41, 0xE3, 0x81, 0x84, 0x42, '¥0'};
74    char str4[4] = {0xE3, 0x81, 0x86, '¥0'};
75    char *r;
```

```
76      r = index(str1, 1);
77      r = index(str2, 2);
78      r = index(str3, 3);
79      r = replace(str3, str4, 3);
80      r = str1;
81  }
```

Translation of technical terms

プログラム	program	関数	function
文字列	string	ポインタ	pointer
C 言語	C programming language	引数	argument
バイト	byte	参照する	refer
最上位ビット	most significant bit	アドレス	address
配列	array	式	expression
16 進数	hexadecimal	固定長	fixed length
10 進表記	decimal notation	割り当てる	allocate
16 進表記	hexadecimal notation		

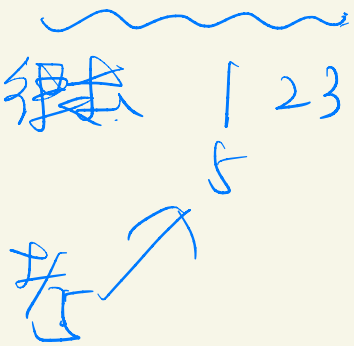
＊　指针

Java. 数组

int[] ⓐ

内存　ⓐ　a+1　a+2　a+3 ……

地址



＊a　值.

不加-　地址.

Author: Aiyi 王

1. *tmp = *string1
   *tmp = *string2
   *tmp = '¥0'

2.

アドレス	r	r+1	r+2	r+3
値	0x41	0x42	0x43	'¥0'

3.

~	r	r+1	r+2	r+3	r+4	r+5	r+6
~	0xE3	0x81	0x84	0xE3	0x81	0x86	'¥0'

4.

アドレス	r	r+1	r+2	r+3	r+4
値	0xE3	0x81	0x84	0x42	'¥0'

5.

アドレス	tmp2	tmp2+1	tmp2+2	tmp2+3	tmp2+4	tmp2+5	tmp2+6
値	0xE3	0x81	0x82	0xE3	0x81	0x86	'¥0'

6. (length_b(string1)+1)

   length_b(tmp1) == 0

7. return tmp2

   break;