■設問１　文字列照合

次の文章を読んで，あとの問いに答えなさい．

テキスト S 中に，単語 W が出現するか否かを判断したい．ここで，各文字列は文字配列として格納されており，S，W の文字数をそれぞれ，|S|，|W| と表すこととする．例えば，S=”sentence” の場合，S[0]=’s’，S[1]=’e’，|S|=8 のようになるということである．

簡単のため，まずは，テキスト S 中に，文字 X が出現するかを判断する「文字照合問題」を考える．この場合，S の先頭の文字から末尾の文字まで，随時，X との比較を行っていけばよい．すなわち，文字位置 i を 0 から |S| まで 1 ずつ増やしていき，随時，S[i]=X となるかを調べるということである．文字 X が出現した場合には，出現した文字位置が返され，出現しなかった場合には，その時点での文字位置である |S| が返されることとなる．よって，例えば，S=”sentence” に対して，X=’t’ の場合に返される値は［　ア　］であり，X=’a’ の場合に返される値は［　イ　］となる．この処理は，一般的な［　ウ　］探索での処理と同一であることからも分かるとおり，実際の処理量（計算時間）は |S| 程度であり，文字数を n として，この計算量は O(n) で表される．

これを踏まえ，テキスト S 中に，単語 W が出現するかを判断する「文字列照合問題」を考える．単純な手法として，文字照合と同様に S 中の各文字位置において W の先頭文字との比較を行い，一致した場合には，単語 W の先頭文字以降との比較を随時行っていく方法が考えられる．すなわち，$i = 0$ から $|S| - |W|$ まで，随時，$j = 0$ として，S[$i + j$]=W[j] となるかを調べ，これが真であった場合，j を 1 ずつ増やしていき |W| まで同様に S[$i + j$]=W[j] となるかを調べるということである．$j = |W|$ となった場合には，文字列が一致したことを意味しているので，文字位置 i を返し，処理を終了する．$j < |W|$ において，S[$i + j$]=W[j] とならなかった場合には，文字列が一致しなかったことを意味しているので，i を 1 増やし，同様の処理を繰り返す．このように，各文字位置 i においては，最大で |W| 回の比較が行われることとなる．よって，この単純な文字列照合手法における計算量は O(　エ　) で表されるが，実際の処理量（計算時間）は，$|S| \times |W|$ 程度になることが予想される．

ここで，W=”abcaba” の文字列照合を行うことを考える．S=”axxxxxxx” の場合，$i = 0$ のとき，$j = 1$ で文字が一致しないことが分かる．よって，次の比較は，$i = 1$，$j = 0$ から行われる．S=”abxxxxxx” の場合，$i = 0$，$j = 2$ で文字が一致しないことが分かる．$j = 1$ では文字が一致していたことが分かっているので，この文字は ’b’ であり，’a’ ではありえない．これを利用して，次の比較は，$i = 1$ から行う必要はなく，$i = 2$，$j = 0$ から行えばよい．同様にして，S=”abcxxxxx” の場合，$i = 0$，$j = 3$ で文字が一致しないことから，次の比較は，$i = 3$，$j = 0$ から行われる．

しかしながら，S=”abcabxxx” の場合には，$i = 0$，$j = 5$ で文字が一致しないからといって，次の比較を $i = 5$ から行うわけにはいかない．なぜなら，$i = 5$ よりも手前の文字列において，一致が発生する可能性があるからである．この文字列の一致を考慮して，次の比較は，$i =$［　オ　］から行う必要がある．この際，前回の比較において，$j = 5$ まで文字が一致していたという事実から，次の比較は $j = 0$ から行う必要はなく，$j =$［　カ　］から行えば十分であることが分かる．このように，次の比較の際に，単純に i を 1 増やし，$j = 0$ とするだけ

でなく，単語 W における文字の重複状況に応じて i，j を適切に管理することにより，文字の比較回数を減少できることが期待される．

このような処理を実現する手法の1つとして［ キ ］法が存在する．この計算量は O(［ ク ］) で表されるが，これにかかる実際の処理量（計算時間）は最大でも 2|S| 程度となることが知られており，単純な手法と比較して，処理を効率化できる可能性がある．

(1) 本文中の［ ア ］～［ ク ］にあてはまる適切な語句や記号，数値を，以下の選択肢から選んで，それぞれ答えなさい．

選択肢
グラフ，線形，二分，CYK，KMP，PL，0，1，2，3，4，5，6，7，8，$\log n$，$\frac{1}{n}$，$\frac{n}{2}$，n，$2n$，$n\log n$，n^2，2^n，$n!$

(2) テキスト S（文字数|S|=18）中に単語 W（文字数|W|=9）が存在するかを［ キ ］法により照合する．

S：araintradraintrain　　W：raintrain

このとき，以下の問いに答えなさい．

(a) 上記単語 W に対するパターン照合テーブル T を作成する．空欄を埋めて以下の表を完成させなさい．

j	0	1	2	3	4	5	6	7	8
W	r	a	i	n	t	r	a	i	n
T	−1	0	0	0	0				
j-T[j]	1	1	2	3	4				

（下書き用）

(b) 以下の表は，［ キ ］法のアルゴリズムに従い解析結果を得るまでの，i，j の値の変化を表している．空欄を埋めて表を完成させ，(あ）～（え）に当てはまる値をそれぞれ答えなさい．

i	0	1	1	1	1	1	1	1	1	(あ)	(う)	9	9	9	9	9	9	9	9	9
j	0	0	1	2	3	4	5	6	7	(い)	(え)	0	1	2	3	4	5	6	7	8	9

※［ キ ］法のアルゴリズム

```
i=0, j=0 から i+j<|S| の間，以下を繰り返し
  W[j]=S[i+j] なら j++
    j=|W| なら i を出力して終了
  W[j]=S[i+j] でなければ
    i=i+j-T[j]
    j>0 ならば j=T[j]
|S| を出力して終了（W が含まれていなかったため）
```

■設問 2　線形連結リスト

以下のデータを順番に読み込み，昇順に並ぶように線形連結リストに格納する．

89, 43, 18, 97, 32

データ 97 までを追加した後のメモリの状態を以下に示す．

address	value	next
—	(head)	0x1050
0x1010	89	0x1070
0x1030	43	0x1010
0x1050	18	0x1030
0x1070	97	NULL

上記にならい，以下の問いに答えなさい．

(1) データを追加する際には，以下の関数が実行される．

```
void insert_cell(struct cell **pointer, int new_value)
{
  struct cell *new_cell;
  new_cell = (struct cell *)malloc(sizeof(struct cell));
  new_cell->value = new_value;
  new_cell->next = *pointer; ①
  *pointer = new_cell; ②
}
```

データ 32 を追加する際における，上記①，②の各処理実行後に対応するメモリの状態の表を以下に記す．空欄を埋めて表を完成させ，（ア）〜（エ）に当てはまる値をそれぞれ答えなさい．

①

address	value	next
—	(head)	0x1050
0x1010	89	0x1070
0x1030	43	0x10　（ア）
0x1050	18	0x10
0x1070	97	NULL
0x1090	32	0x10　（イ）

②

address	value	next
—	(head)	0x1050
0x1010	89	0x1070
0x1030	43	0x10
0x1050	18	0x10　（ウ）
0x1070	97	NULL
0x1090	32	0x10　（エ）

(2) データを削除する際には，以下の関数が実行される．

```
void delete_cell(struct cell **pointer)
{
   struct cell *target;
   target = *pointer;
  *pointer = target->next;
   free(target);
}
```

上記 (1) の結果に対して，データ 43 を削除した後におけるメモリの状態の表を以下に記す．空欄を埋めて表を完成させ,（ア）に当てはまる値を答えなさい．

address	value	next
——	(head)	0x10
0x1010	89	0x10
0x1030	——	——
0x1050	18	0x10
0x1070	97	NULL
0x1090	32	0x10　（ア）

■設問 3　二分探索

昇順にソートされている以下の 9 件の整数データが配列 a[] に格納されており，これから任意のデータを二分探索により検索することを考える．

a[]： 19　21　33　38　59　70　71　76　98

```
int binsearch(int x, int *a, int n)
{
    int m, l=0, r=n-1;

    if (x<=a[l]) return 0;
    if (x>a[r]) return n;
    while (r-l>1) {
        m=(r+l)/2;
        if (x<=a[m]) r=m;
        else l=m;
    }
    return r;
}
```

二分探索は左に示す関数 binsearch() により実現されるものとする．この関数は，検索対象となるデータ x，配列 a[]，その要素数 n=9 を引数として呼び出される．よって，例えば，76 を引数 x として与えた場合，変数 r の値は，8 → 7 と変化し，最終的に返される値（戻り値）は 7 となる．

同様にして，以下の各値を引数 x として与えたとき，変数 r の値の変化と戻り値をそれぞれ答えなさい．

(1) 21　　(2) 60　　(3) 100

■設問４　循環・重連結リスト

循環・重連結リストについて，下記ソースコードにおける各関数を実行する．このとき，以下の問いに答えなさい．

```
typedef struct Node {
    int num;
    struct Node *left, *right;
} node;

int insert_cdl_list(int x) {
    node *q, *p =
      (node *)malloc(sizeof(node));
    if(p == NULL)
        return 0;
    p -> num = x;

    [ ①                    ]

    return 1;
}

int insert_left(node *p, int x) {
    node *q =
      (node *)malloc(sizeof(node));
    if(q == NULL)
        return 0;
    q -> num = x;

    [ ②                    ]

    return 1;
}
```

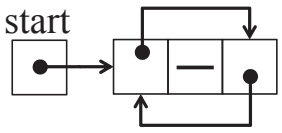


図 3-1

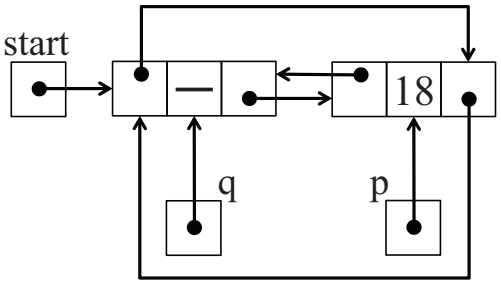


図 3-2

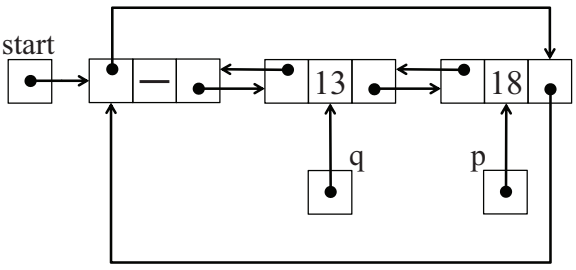


図 3-3

(1) 図 3-1 に示す循環･重連結リストに対して，上記ソースコードにおける insert_cdl_list 関数を insert_cdl_list(18) として実行した．その途中，［①］の処理を終えた段階で，図 3-2 に示す循環・重連結リストを得ることができた．このとき，［①］に入るソースコードとして正しいものを（ア）～（カ）からすべて選び，答えなさい．

（ア）
```
p -> right = start;
start -> left = p;
p -> left = q;
q -> right = p;
q = start -> left;
```

（イ）
```
p -> left = q;
q = start -> left;
p -> right = start;
q -> right = p;
start -> left = p;
```

（ウ）
```
q = start -> left;
p -> right = start;
p -> left = q;
q -> right = p;
start -> left = p;
```

（エ）
```
q = start -> left;
q -> right = p;
p -> left = q;
p -> right = start;
start -> left = p;
```

（オ）
```
start -> left = p;
q = start -> left;
p -> right = start;
q -> right = p;
p -> left = q;
```

（カ）
```
p -> right = start;
p -> left = q;
q -> right = p;
q = start -> left;
start -> left = p;
```

(2) 上記 (1) で得られた循環･重連結リストに対して，上記ソースコードにおける `insert_left` 関数を `insert_left(r, 13)` として実行した．ここで，`r` はデータ 18 が格納されているノードの先頭アドレスを表している．その途中，［②］の処理を終えた段階で，図 3-3 に示す循環・重連結リストを得ることができた．このとき，［②］に入るソースコードとして正しいものを（ア）～（カ）からすべて選び，答えなさい．

（ア）
```
p -> left -> right = q;
q -> left = p -> left;
p -> left = q;
q -> right = p;
```

（イ）
```
p -> left -> right = q;
p -> left = q;
q -> left = p -> left;
q -> right = p;
```

（ウ）
```
p -> left = q;
p -> left -> right = q;
q -> right = p;
q -> left = p -> left;
```

（エ）
```
p -> left = q;
q -> right = p;
p -> left -> right = q;
q -> left = p -> left;
```

（オ）
```
q -> left = p -> left;
p -> left -> right = q;
q -> right = p;
p -> left = q;
```

（カ）
```
q -> right = p;
q -> left = p -> left;
p -> left = q;
p -> left -> right = q;
```

■設問５　ハッシュ法

ハッシュ法（hashing）により，文字列で表されるキーを，適当な大きさを持つ配列に格納する．キーは互いに重複しないものとする．3 文字以上で構成される文字列 s をキーとして，配列の要素数 N，ハッシュ関数 H，別のハッシュ関数 H_i を以下のように定義する．

$$N = 1021 \qquad H(s) = s[0] + 3 \times s[2] \qquad H_i(s) = 5 \times s[1] + strlen(s)$$

ここで，$s[0], s[1], s[2]$ は，それぞれ，文字列 s における 1 文字目，2 文字目，3 文字目の文字コードを表しており，$strlen(s)$ は文字列 s の長さ（文字数）を表している．なお，各文字コードは，10 進数で以下のようになっている．

A = 65　B = 66　C = 67　D = 68　E = 69　...　H = 72　...　O = 79　P = 80

Q = 81　R = 82　S = 83　T = 84　U = 85　...　X = 88　Y = 89　Z = 90

例えば，キーとして文字列 “XYZ” が与えられたとき，その一次インデクス値（primary index value），および，仮にこれが衝突（collision）していた場合に，別のハッシュ関数により計算される増分の値は以下のとおりである．

$$H(\text{“XYZ”}) = 88 + 3 \times 90 = 358 \qquad H_i(\text{“XYZ”}) = 5 \times 89 + 3 = 448$$

いま，配列において，以下の要素番号の場所には，互いに異なるキーが，すでに格納済みである．

265〜270，320〜325，665〜670，745〜750

ここでは，これを初期状態と呼ぶ．このとき，以下の問いに答えなさい．

(1) 初期状態の配列から特定のキーを探索する．

(a) 要素番号 270 の場所に，キー “HEAD” が格納されている．これを線形探査法（linear probing）に従い探索する際，走査することになる要素番号を順番にすべて答えなさい．

(b) 要素番号 669 の場所に，キー “BODY” が格納されている．これをダブルハッシュ法（double hashing）に従い探索する際，走査することになる要素番号を順番にすべて答えなさい．

(2) 初期状態の配列において，それぞれの手法で，以下のキーを，この順番で格納する．

“EASY”，“HURRY”，“CUTE”

(a) 線形探査法に従うとき，格納される要素番号をそれぞれ答えなさい．

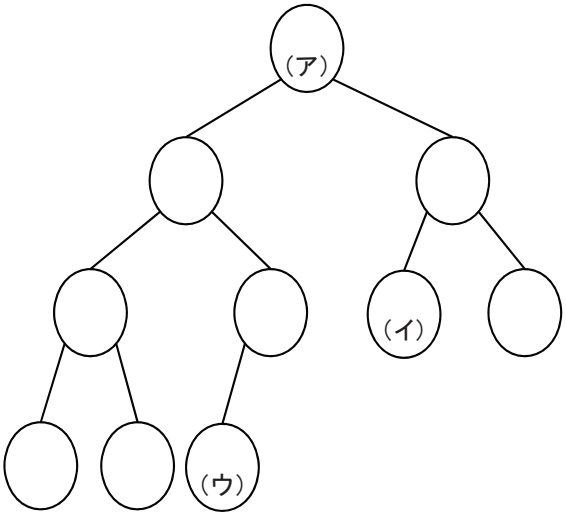
(b) ダブルハッシュ法に従うとき，格納される要素番号をそれぞれ答えなさい．

■設問 6　ヒープソート

10 個の整数データからなる次の配列について，以下の問いに答えなさい．ただし，当該配列において，0 番目の要素は使用せず，1 番目以降の要素として順に 10 個のデータが格納されているものとする．

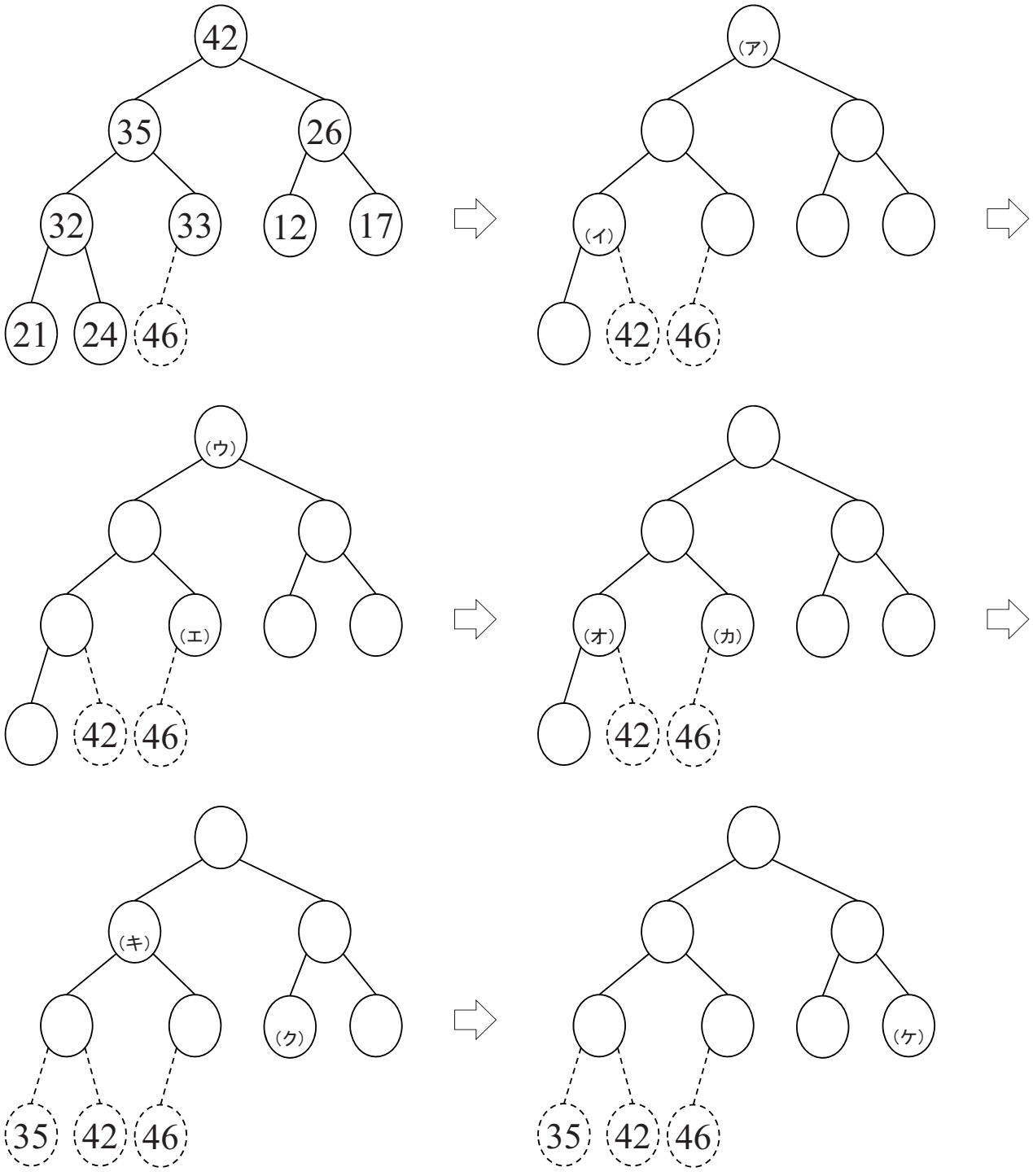
33，24，12，32，35，42，26，21，46，17

(1) 以下の図は，上記配列に対応する完全二分木を表している．空欄になっている各ノードを適当な数値で埋め，（ア）〜（ウ）に入る数値をそれぞれ答えなさい．



(2) 上記 (1) の完全二分木を下降修復の繰り返しによりヒープ化した結果を配列の形で答えなさい．

(3) 上記 (2) で得られたヒープに対して，ヒープソートによりデータを昇順に整列することを考える．このときの途中経過として，5 回分の置換結果を以下に示している．空欄を埋めて図を完成させ,（ア）～（ケ）に入る数値をそれぞれ答えなさい．なお，破線で記したノードは値が確定していることを表している．



■設問 7　マージソート

自然マージソートにより，ファイル F に格納されている以下の 13 件のデータを昇順にソートする．

F：26, 12, 34, 46, 11, 28, 21, 20, 45, 40, 44, 15, 23

(1) ファイル F のデータを連（run）で分割した結果を答えなさい．ただし，連の分割はスラッシュ（/）により明示すること．

(2) それぞれの連をファイル A, B へ分配，すなわち，交互に格納していく．最初の連をファイル A に格納するものとしたとき，ファイル A, B それぞれにおける，分配した結果を答えなさい．

(3) ファイル A, B において，再度，連を作成し，それぞれを順にマージし，ファイル F に格納する．このとき，ファイル F の結果を答えなさい．

(4) 同様に，分配，マージを繰り返していき，ソートを行う．このとき，最終的なソート結果を答えなさい．ただし，上記 (3) 以降，最終的な結果に至るまでの，分配，マージの結果をすべて示すこと．

F:	（上記 (3) の結果）												
A:													
B:													
F:													
A:													
B:													
F:													
A:													
B:													
F:													
A:													
B:													
F:													

（下書き用）