次の問８は必須問題です。必ず解答してください。

問８　ハフマン符号化を用いた文字列圧縮に関する次の記述を読んで，設問１～３に答えよ。

“A”～“D”の４種類の文字から成る文字列をハフマン符号化によって圧縮する。ハフマン符号化では，出現回数の多い文字には短いビット列を，出現回数の少ない文字には長いビット列を割り当てる。ハフマン符号化による文字列の圧縮手順は，次の(1)～(4)のとおりである。

(1)　文字列中の文字の出現回数を求め，出現回数表を作成する。例えば，文字列“AAAABBCDCDDACCAAAAA”（以下，文字列αという）中の文字の出現回数表は，表１のとおりになる。

表１　文字列α中の文字の出現回数表

文字	A	B	C	D
出現回数	10	2	4	3

(2)　文字の出現回数表に基づいてハフマン木を作成する。

ハフマン木の定義は，次のとおりである。

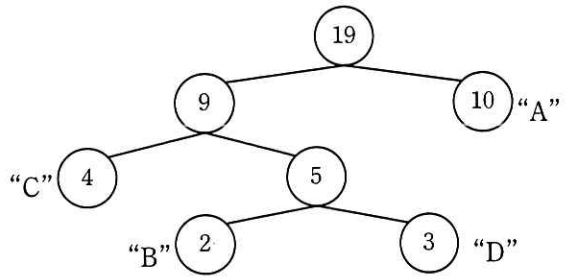
・　節と枝で構成する二分木である。

・　親である節は，子である節を常に二つもち，子の節の値の和を値としてもつ。

・　子をもたない節（以下，葉という）は文字に対応し，出現回数を値としてもつ。

・　親をもたない節（以下，根という）は，文字列の文字数を値としてもつ。

文字列αに対応するハフマン木の例を，図１に示す。



注記　丸の中の数値は各節がもつ値を表す。"A"～"D"は葉に対応する文字を表す。

図1　文字列αに対応するハフマン木の例

ハフマン木は，次の手順で配列によって実現する。

① 節の値を格納する1次元配列を用意する。

② 文字の出現回数表に基づいて，各文字に対応する葉の値を，配列の先頭の要素から順に格納する。

③ 親が作成されていない節を二つ選択し，選択した順に左側の子，右側の子とする親の節を一つ作成する。この節の値を，配列中で値が格納されている最後の要素の次の要素に格納する。節の選択は節の値の小さい順に行い，同じ値をもつ節が二つ以上ある場合は，配列の先頭に近い要素に値が格納されている節を選択する。

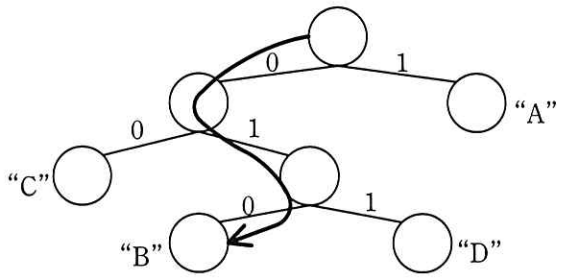
④ 親が作成されていない節が一つになるまで③を繰り返す。

(3) ハフマン木から文字のビット列（以下，ビット表現という）を次の手順で作成する。

① 親と左側の子をつなぐ枝に 0，右側の子をつなぐ枝に 1 の値をもつビットを割り当てる。

② 文字ごとに根から対応する葉までたどったとき，枝のビット値を順に左から並べたものを各文字のビット表現とする。

図 2 に示すとおり，根から矢印のようにたどると，文字列αの文字"B"のビット表現は 010 となる。



注記　線分は枝を表し，枝の上の数値は各枝のビット値を表す。

図２　文字列αにおける文字“B”のビット表現の作成例

(4)　文字列の全ての文字を(3)で得られたビット表現に置き換えて，ビット列を作成する。

設問１　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

文字列“ABBBBBBBCCCDD”を，ハフマン符号化を用いて表現する。各文字とビット表現を示した表は［ a ］である。ハフマン符号化によって圧縮すると，文字“A”～“D”をそれぞれ２ビットの固定長で表現したときの当該文字列の総ビット長に対する圧縮率は［ b ］となる。ここで，圧縮率は次式で計算した値の小数第３位を四捨五入して求める。

$$\text{圧縮率} = \frac{\text{ハフマン符号化によって圧縮したときの総ビット長}}{\text{２ビットの固定長で表現したときの総ビット長}}$$

aに関する解答群

ア

文字	A	B	C	D
ビット表現	010	1	00	011

イ

文字	A	B	C	D
ビット表現	010	0	01	111

ウ

文字	A	B	C	D
ビット表現	100	0	101	11

エ

文字	A	B	C	D
ビット表現	100	1	00	01

bに関する解答群

ア　0.77　　イ　0.85　　ウ　0.88　　エ　0.92

設問2　ハフマン木を作成するプログラム1の説明及びプログラム1を読んで，プログラム1中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔プログラム1の説明〕

(1) 四つの1次元配列 parent，left，right 及び freq の同じ要素番号に対応する要素の組み（以下，要素組という）によって，一つの節を表す。要素番号は0から始まる。四つの配列の大きさはいずれも十分に大きく，全ての要素は−1で初期化されている。

(2) 図3に，図1に示したハフマン木を表現した場合の各配列の要素がもつ値を示す。配列 parent には親，配列 left には左側の子，配列 right には右側の子を表す要素組の要素番号がそれぞれ格納され，配列 freq には節の値が格納される。節が葉のとき，配列 left と配列 right の要素の値は，いずれも −1 である。図3では，要素番号0〜3の要素組が，順に文字“A”〜“D”の葉に対応している。節が根のとき，配列 parent の要素の値は−1である。

配列名	要素番号						
	0	1	2	3	4	5	6
parent	6	4	5	4	5	6	−1
left	−1	−1	−1	−1	1	2	5
right	−1	−1	−1	−1	3	4	0
freq	10	2	4	3	5	9	19

注記　矢印●⟶は，始点，終点の二つの要素組に対応する節が子と親の関係にあることを示す。

図3　図1に示したハフマン木を表現する四つの配列

(3) 副プログラム Huffman は，次の①〜⑤を受け取り，ハフマン木を表現する配列を作成する。

① 葉である節の個数 size

② 初期化された配列 parent

③ 初期化された配列 left

④ 初期化された配列 right

⑤ 初期化された後，文字の出現回数が要素番号 0 から順に格納された配列 freq

(4) 副プログラム SortNode は，親が作成されていない節を抽出し，節の値の昇順に整列し，節を表す要素組の要素番号を順に配列 node に格納し，その個数を変数 nsize に格納する。行番号 19～24 で親が作成されていない節を表す要素組の要素番号を抽出し，行番号 25 で節の値の昇順に整列する。

(5) 副プログラム Sort（プログラムは省略）は，節を表す要素組の要素番号の配列 node を受け取り，要素番号に対応する要素組が表す節の値が昇順となるように整列する。節の値が同じときの順序は並べ替える直前の順序に従う。

(6) 副プログラム Huffman，SortNode 及び Sort の引数の仕様を，表 2～4 に示す。

表 2　副プログラム Huffman の引数の仕様

引数	データ型	入出力	説明
size	整数型	入力／出力	節の個数
parent[]	整数型	入力／出力	節の親を表す要素組の要素番号を格納した配列
left[]	整数型	入力／出力	節の左側の子を表す要素組の要素番号を格納した配列
right[]	整数型	入力／出力	節の右側の子を表す要素組の要素番号を格納した配列
freq[]	整数型	入力／出力	節の値を格納した配列

表 3　副プログラム SortNode の引数の仕様

引数	データ型	入出力	説明
size	整数型	入力	節の個数
parent[]	整数型	入力	節の親を表す要素組の要素番号を格納した配列
freq[]	整数型	入力	節の値を格納した配列
nsize	整数型	出力	配列 node 中の，整列対象とした節の個数
node[]	整数型	出力	節の値の昇順に整列した，親が作成されていない節を表す要素組の要素番号を格納した配列

表4　副プログラム Sort の引数の仕様

引数	データ型	入出力	説明
freq[]	整数型	入力	節の値を格納した配列
nsize	整数型	入力	配列 node 中の，整列対象の節の個数
node[]	整数型	入力／出力	節を表す要素組の要素番号を格納した配列

〔プログラム 1〕

(行番号)

```
1   ○副プログラム: Huffman(整数型: size, 整数型: parent[], 整数型: left[],
                          整数型: right[], 整数型: freq[])
2   ○整数型: i, j, nsize
3   ○整数型: node[]
4   ・SortNode(size, parent, freq, nsize, node)
5   ■  [ c ]
6   |  ・i ← node[0]      /* 最も小さい値をもつ要素組の要素番号 */
7   |  ・j ← node[1]      /* 2番目に小さい値をもつ要素組の要素番号 */
8   |  ・left[size] ← i
9   |  ・right[size] ← j
10  |  ・freq[size] ← freq[i] + freq[j]  /* 子の値の合計 */
11  |  ・parent[i] ←  size   /* 子に親の節の要素番号を格納 */
12  |  ・parent[j] ←  size   /* 子に親の節の要素番号を格納 */
13  |  ・size ← size + 1
14  |  ・SortNode(size, parent, freq, nsize, node)
15  ■

16  ○副プログラム: SortNode(整数型: size, 整数型: parent[], 整数型: freq[],
                          整数型: nsize, 整数型: node[])
17  ○整数型: i
18  ・nsize ← 0
19  ■ i: 0, i < size, 1
20  |▲ [ d ]
21  |  ・node[nsize] ← i
22  |  ・nsize ← nsize + 1
23  |▼
24  ■
25  ・Sort(freq, nsize, node)
```

c，dに関する解答群

ア　nsize ≧ 0　　イ　nsize ≧ 1　　ウ　nsize ≧ 2

エ　parent[i] ＜ 0　　オ　parent[i] ＞ 0　　カ　size ≦ nsize

キ　size ≧ nsize

設問3　ハフマン木から文字のビット表現を作成して表示するプログラム 2 の説明及びプログラム 2 を読んで，プログラム 2 中の［　　］に入れる正しい答えを，解答群の中から選べ。

〔プログラム 2 の説明〕

(1)　ビット表現を求めたい文字に対応する葉を表す要素組の要素番号を，副プログラム Encode の引数 k に与えて呼び出すと，ハフマン木から文字のビット表現を作成して表示する。

(2)　副プログラム Encode の引数の仕様を，表 5 に示す。

表 5　副プログラム Encode の引数の仕様

引数	データ型	入出力	説明
k	整数型	入力	節を表す要素組の要素番号
parent[]	整数型	入力	節の親を表す要素組の要素番号を格納した配列
left[]	整数型	入力	節の左側の子を表す要素組の要素番号を格納した配列

(3)　副プログラム Encode は，行番号 2 の条件が成り立つとき，副プログラム Encode を再帰的に呼び出す。これによって，ハフマン木を葉から根までたどっていく。

(4)　根にたどり着くと次は葉に向かってたどっていく。現在の節が親の左側の子のときは 0 を，右側の子のときは 1 を表示する。

(5)　関数 print は，引数で与えられた文字列を表示する。

〔プログラム2〕

(行番号)

```
1   ○副プログラム: Encode(整数型: k, 整数型: parent[], 整数型: left[])
2   ▲ [ e ]
3   ｜ ・Encode(parent[k], parent, left)
4   ｜ ▲ [ f ]
5   ｜ ｜ ・print("0")    /* 0を表示する */
6   ｜ ｜ ・print("1")    /* 1を表示する */
7   ｜ ▼
8   ▼
```

eに関する解答群

ア　k ≧ 0　　イ　left[k] = −1　　ウ　left[k] ≧ 0

エ　parent[k] = −1　　オ　parent[k] ≧ 0

fに関する解答群

ア　left[k] = k　　イ　left[parent[k]] = k

ウ　parent[k] = k　　エ　parent[left[k]] = k