問題

図 1 はマージソートの様子を再帰木で示したものである。図 1 に示す再帰木の末端がマージソートの分割直後の状態とする時、かっこの番号に適切な数値を入力して、マージソートの merge 関数の分割操作後の再帰木を完成せよ。再帰木の一部の状態は虫食いにしてあるため、アルゴリズムから動作の状態を推測すること。マージソートのアルゴリズムは参考にしてよい。

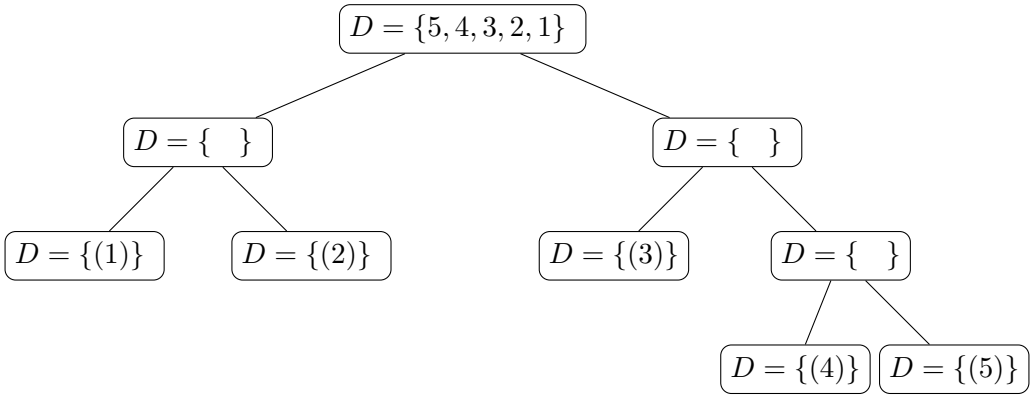

図 1　マージソートの merge 関数実行時の再帰木

アルゴリズム 7.3　マージソート

```
入力：サイズ n の配列 D[0], D[1], ..., D[n-1]
mergesort(D, left, right) {
  mid=(left + right)/2;                                  ---(1)
  if (left < mid) mergesort(D, left, mid);               ---(2)
  if (mid+1 < right) mergesort(D, mid+ 1, right);        ---(2)
  merge(D, left, mid, right);                            ---(3)
}
//mergesort(D, 0, n-1)を実行することにより入力全体のソートが実行される.
```

図 2　マージソートのアルゴリズム (mergesort 関数)

アルゴリズム 7.4　関数 merge

```
merge(D,left,mid,right)  {
  x=left; y=mid+1;
  for (i=0; i<=right-left; i=i+1)  {
    if (x==mid+1)  { M[i]=D[y]; y=y+1; }          //左のソート列が空の場合
    else if (y==right+1)  { M[i]=D[x]; x=x+1; } //右のソート列が空の場合
    else if (D[x]<=D[y])  { M[i]=D[x]; x=x+1; }
                                        //左のソート列の最小値が小さい場合
    else { M[i]=D[y]; y=y+1; }          //右のソート列の最小値が小さい場合
  }

  for (i=left; i<=right; i=i+1)  { D[i]=M[i]; }  //配列Mを配列Dにコピー
}
```

図 3　マージソートのアルゴリズム (merge 関数)