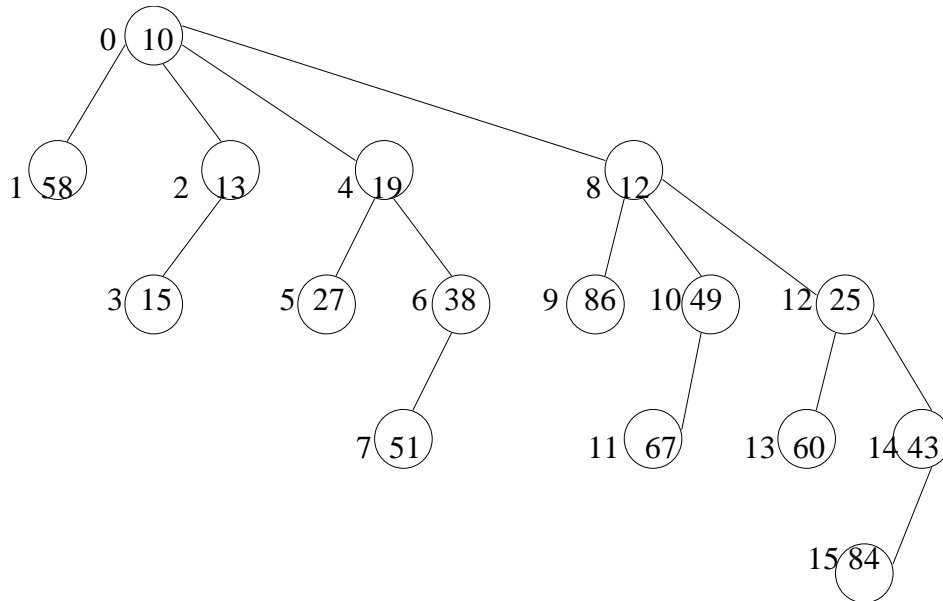


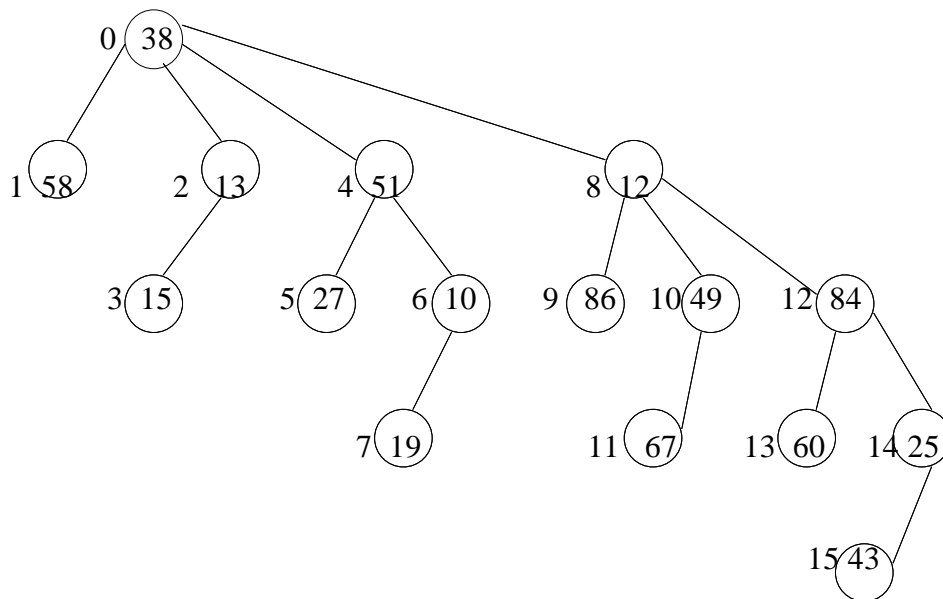
Question 1. [20 marks] We design a sorting algorithm, similar to heap sort, based on a binomial queue as described below. For simplicity we assume the number of keys to be sorted, n , to be a power of 2. The following example is a binomial queue of size 16.



The index of each node corresponds to the location of the key in a one-dimensional array a that stores the keys as shown below. Note odd-numbered nodes are leaves.

$a = (10 \ 58 \ 13 \ 15 \ 19 \ 27 \ 38 \ 51 \ 12 \ 86 \ 49 \ 67 \ 25 \ 60 \ 43 \ 84)$
 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Once a binomial queue is made, the rest is to swap $a[0]$ and $a[n-1]$ and recover the binomial queue for the portion $a[0 .. n-2]$, then swap $a[0]$ and $a[n-2]$ and recover $a[0 .. n-3]$, and so on. To make a binomial queue from the tree with random keys in the nodes, we start from node 14, going 12, 10, ..., 0, performing the sift up operation at each node. The above binomial queue is formed from the following initial tree.



A C program for the binomial queue sort, abbreviated bq-sort is given below with some vacant portion. The binomial queue is formed in a bottom-up fashion. The siftup function selects the minimum $a[m]$ among its children $a[p+1]$, $a[p+2]$, $a[p+4]$, ..., and swap $a[p]$ and $a[m]$ and calls siftup at m if necessary, going down the tree.

```

1. int i,ii,n,t,k;
2. int a[100000],y,z;
3. int even(x) /* This function tests if x is even */
4. int x;
5. { if((x/2)*2==x)return 1; else return 0;}
6. swap(i,j) /* swapping function */
7. int i,j;
8. { int t; t=a[i]; a[i]=a[j]; a[j]=t;}
9. siftup(p,q) /* function siftup */
10. int p, q;
11. {
12.     int j, k, m, r;
13.     y=a[p]; k=1;
14.     z=a[p]; j=p+1; if(p==0) r=n; else r=p;
15.     m=p;
16.     while (even(r)&&(j<=q)) {
17.         if (a[j]<z){ .....; .....;}
18.         j=j+k; k=2*k; r=r/2;
19.     } /* z=a[m] is the minimum among the children of a[p] */
20.     if (z<y) { swap(p,m); if (even(m)) siftup(m,q); }
21. } /* end of siftup */
22. heap(n)
23. int n;
24. { /* Build a binomial queue */
25.     for (i=n-2; i>=0; i=i-2) {
26.         siftup(i,n);
27.     }
28.     /* a binomial queue has been made */
29.     for (i=n-1; i>=1; i--) { /* Actual sorting process starts */
30.         swap(0,i);
31.         siftup(0,i-1);
32.     }
33. }
34. main()
35. { printf("input size \n");
36.     scanf("%d",&n);
37.     for (i=0;i<=n-1;i++) a[i]=rand()% 100;
38.     for (i=0;i<=n-1;i++) printf("%d ", a[i]); printf("\n");
39.     t=clock();
40.     heap(n);
41.     for (i=0;i<=10;i++) printf("%d ",a[i]);
42.     printf("\n");
43.     printf("time= %d millisec\n",(clock()-t)/1000);
44. }
```

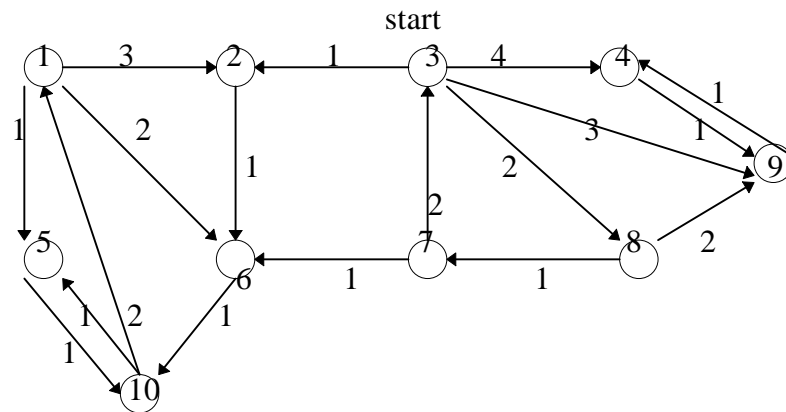
(A) Fill the vacant portion in line 17. This algorithm sorts the keys in decreasing order. How do you modify the algorithm to sort keys in increasing order? 6 marks

(B) For our example of binomial queue, perform lines 30 and 31 with $i=15$ and 14 , and show the results. Show the picture of each tree and show the path through which the root element traversed down. 6 marks.

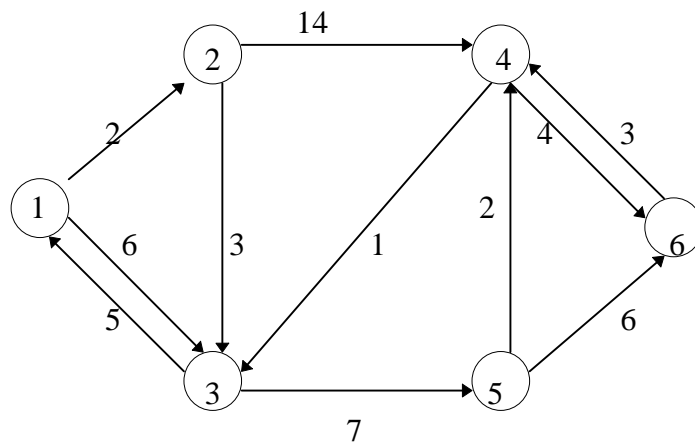
(C) Show that the computing time of this algorithm is $O(n(\log n)^2)$. 8 marks.

Question 2. [20 marks] Insert the sequence $L = (38\ 58\ 13\ 15\ 51\ 27\ 10\ 19\ 12\ 86\ 49\ 67)$ into an AVL tree. Show the picture of the tree after each insertion and the rotation if caused by the insertion. To show a rotation, show the subtrees A, B, C, (and D if any) as shown in the lecture notes.

Question 3. [20 marks] Trace Tarjan's algorithm for strongly connected (sc) components for the following graph. For trace, show the depth-first spanning tree with low link numbers, and sc-components in the order given by the algorithm. The names of vertices are given by integers in or boundaries of circles. The integers attached close to edges are to show the order in which the algorithm examines the edges from each vertex.



Question 4. [20 marks] A directed graph is given as follows:



(A) Trace Dijkstra's algorithm with this graph. 10 marks

(B) Trace Floyd's algorithm with this graph. 10 marks

Choose Question 5 or 6 for the fifth question to answer.

Question 5. [20 marks] The pattern and text are given as follows:

pat = ababc and text = ababaababc.

The algorithm for computing the shift heuristic h is given as follows in which print statements are given for trace.

```
/* input/output statements are omitted */
t=0; h[1]=0;
for(i=2; i<=m; i++){
    while((t>0)&&(pat[i-1]!=pat[t])) t=h[t];
    t++;
    printf("i, t %d %d pat[i] pat[t] %c %c ",i, t, pat[i], pat[t]);
    if (pat[i]!=pat[t])h[i]=t; else h[i]=h[t];
    for(j=1;j<=i;j++)printf("%d ",h[j]); printf("\n");
}
for(i=1;i<=m; i++) printf("%d ",h[i]);
```

The trace for h with the above pattern is given as follows:

```
input pat = a b a b c
i, t 2 1 pat[i] pat[t] b a 0 1
i, t 3 1 pat[i] pat[t] a a 0 1 0
i, t 4 2 pat[i] pat[t] b b 0 1 0 1
i, t 5 3 pat[i] pat[t] c a 0 1 0 1 3
0 1 0 1 3
```

The trace of the main matching process is informally shown as follows:

```
a b a b a a b a b c
a b a b c          h[5]=3
  a b a b c       h[4]=1
    a b a b c     12 comparisons made
```

Following this example, trace the KMP algorithm for h and main matching process with pat = ababac and text = abababaababac.

Question 6. [20 marks] Trace Euclid's algorithm with a = 29 and b = 11. From the trace obtain $29^{-1} \pmod{11}$ and $11^{-1} \pmod{29}$.