Scilab Textbook Companion for Fundamentals Of Data Structure In C by S. Sahni , S. Anderson-freed And E. Horowitz[1]

Created by
Aakash Yadav
B-TECH
Computer Engineering
Swami Keshvanand Institute of Technology
College Teacher
Richa Rawal
Cross-Checked by
Lavitha Pereira

July 31, 2019

Book Description

Title: Fundamentals Of Data Structure In C

Author: S. Sahni , S. Anderson-freed And E. Horowitz

Publisher: University Press (India) Pvt. Ltd., New Delhi

Edition: 2

Year: 2008

ISBN: 9788173716058

Scilab numbering policy used in this document and the relation to the above book.

Exa Example (Solved example)

Eqn Equation (Particular equation of the above book)

AP Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

Contents

List of Scilab Codes

Chapter 1

Basic concepts

Scilab code Exa 1.1 example

```
1  //to do sorting of nos. contained in a list
2  function []= sorting (a)
3      i =1;
4      [j ,k]= size (a);
5      j=i;
6      for i =1:k -1
7          for j=i:k
8              if a(i)>a(j) then
9              z=a(i);
10             a(i)=a(j);
11             a(j)=z;
12          end
13      end
14  end
15  for i =1:k
16      disp (a(i));
17  end
18
19  funcprot (0);
20  endfunction
21      //callin  routine
```

```
22      a=[5 7 45 23 78]
23      sort=sorting(a)
```

Scilab code Exa 1.2 example

```
1  // to do binary search..
2  function []=search(a)
3      i=1;
4      [j,k]=size(a);
5              for i=1:k
6                  if z==a(i) then
7                      printf(" \nFOUND and index no. is
                           =%d\t",i);
8                  end
9              end
10          funcprot(0);
11 endfunction
12 //callin routine
13 a=[5 7 45 28 99]
14 z=45
15 binary=search(a)
```

Scilab code Exa 1.3 example

```
1  // to do binary search..
2  function []=search(a)
3      i=1;
4      [j,k]=size(a);
5              for i=1:k
6                  if x==a(i) then
7                      printf(" \nFOUND and index no. is
                           =%d\t",i);
8                  end
```

```
9           end
10       funcprot(0);
11 endfunction
12 //callin routine
13 a=[5 7 45 28 99]
14 x=45
15 binary=search(a)
```

Scilab code Exa 1.4 example

```
1 // example 1.4
2 // permutation of a string or character array...
3 clear;
4 clc;
5 x=['a' 'b' 'c' 'd']
6 printf("\npossible permutation of given string are\n
     ");
7 y=perms(x);
8 disp(y);
```

Scilab code Exa 1.5 example

```
1 //      example 1.5
2 //   ADT(Abstract Data type) defination of natural
     number.
3  function[]=ADT(x)
4      printf("ADT natural no. is ");
5  printf("\nOBJECTS: an ordered subrange of the
      integers starting at zero and ");
6      printf("ending at the maximun integer (INT_MAX)
          on the computer");
7      INT_MAX=32767;
8      if x==0             //NaturalNumberZero()
```

```
9          printf("\n",0);
10         end
11         if x==INT_MAX then        //
               NaturalNumberSuccessor(x)
12             printf("\nans. is=%d",x);
13         else
14             printf("\nans. is=%d",x+1);
15         end
16     endfunction
17     //callin routine
18     x=56
19     y=ADT(x);
```

Scilab code Exa 1.6 example

```
1  //function abc accepting only three simple variables
       given the function has
2   //only fixed sace requirement..
3   function []=abc(a,b,c)
4       x= a+b+c*c+(a+b-c)/(a+b)+4.00;
5       disp(x);
6       funcprot(0);
7   endfunction
8   ....//calling routine
9   a=[1],b=[2],c=[3]
10  abc(a,b,c)
```

Scilab code Exa 1.7 example

```
1  // To add a list of no. using array.
2  function []=add(a)
3      result=sum(a);
4
```

```
5        printf("addition  of  no.  on  the  list  is=%d",
            result);
6        funcprot(0);
7   endfunction
8   //calling  routine
9   a=[5  2  7  8  9  4  6]
10  r=add(a)
```

Scilab code Exa 1.8 example

```
1   clear;
2   clc;
3   printf("\n  Example  1.8");
4   a=[2;5;4;64;78]
5   i=1;
6   x=1;............//initialising  sum  equals  to  one.
7   c=1;.............//  initialising  count  equals  to
        one.
8   while  i<6
9        c=c+a(i);........//sum
10       x=x+1;..............////step  count
11       i=i+1;
12  end
13  printf("\n  no.  in  the  list  are  a=")
14  printf("\n  %d",a);
15  printf("\n  sum  is=%d",(c-1));
16  printf("\n  count  is=%d",(x-1));
```

Scilab code Exa 1.9 example

```
1   clear;
2   clc;
3   printf("\n  Example  1.9");
```

```
4  a=[1 2 3;4 5 6];
5  b=[7 8 9;10 11 12];
6  x=matrix(a,3,2);........//no. of rows=3,no. of col.
       =2.
7  y=matrix(b,3,2);........//no, of rows=3,no. of col
       .=2.
8  printf("matrix x=");
9  disp(x);
10 printf("matrix y=");
11 disp(y);
12 [p,q]=size(x);
13 i=1;
14 j=1;
15 c=1;
16 for i=1:p
17     for j=1:q
18         z(i,j)=x(i,j)+y(i,j);..........//summing two
                matrices
19         c=c+1;...........//step count
20     end
21 end
22 printf("\n Resultant matrix after addition =");
23 disp(z);.............//displayin sum of two matrices
       .
24 printf("\n step count is=%d",(c-1));
```

Scilab code Exa 1.10 `example`

```
1 clear;
2 clc;
3 printf("\n Example 1.10");
4 // function to sum a list of numbers.
5 function []=add()
6     printf("\n no. in the list are");
7     disp(a);
```

```
8       x=sum(a);
9       printf("\n Result=%d",x);
10      funcprot(0);
11      endfunction
12      //calling routine.
13      a=[2 5 6 7 9 1 6 3 7 45]
14      add()
```

Scilab code Exa 1.11 example

```
1  clear;
2  clc;
3  printf("\n Example 1.11");
4  // Matrix addition.
5  a=[1 2 3;4 5 6];
6  b=[7 8 9;10 11 12];
7  x=matrix(a,3,2);........//no. of rows=3,no. of col.
      =2.
8  y=matrix(b,3,2);........//no, of rows=3,no. of col
      .=2.
9  printf("matrix x=");
10 disp(x);
11 printf("matrix y=");
12 disp(y);
13 [p,q]=size(x);
14 i=1;
15 j=1;
16 for i=1:p
17     for j=1:q
18         z(i,j)=x(i,j)+y(i,j);..........//summing two
                matrices
19     end
20 end
21 printf("\n Resultant matrix after addition =");
22 disp(z);.............//displayin sum of two matrices
```

.

Scilab code Exa 1.12 `example`

```
1  clear ;
2  clc ;
3  printf ("Example 1.12");
4  // [BIG "oh"] f(n)=O(g(n)). (big oh notation).
5  printf ("\n \n 3n+2=O(n) as 3n+2<=4n for all n>=2.");
6  printf ("\n \n 3n+3=O(n) as 3n+3<=4n for all n>=3.")
       ;.............// O(n) is called linear.
7  printf ("\n \n 3n+2=O(n) as 100n+6<=101n for all n
       >=10.");
8  printf ("\n \n 10n^2+4n+2=O(n^2) as 10n^2+4n+2<=11n^2
        for n>=5.");..........//O(n^2) is called
       quadratic.
9  printf ("\n \n 1000n^2+100n−6=O(n^2) as 1000n^2+100n
       −6<=1001n^2 for n>=100.");
10  printf ("\n \n 6*2^n+n^2<=7*2^n for n>=4");
11  printf ("\n \n 3n+3=O(n^2) as 3n+3<=3n^2 for n>=2");
12  printf ("\n \n 10n^2+4n+2=O(n^4) as 10n^2+4n+2<=10n^4
        for n>=2.");
13  printf ("\n \n 3n+2 is not O(1) as 3n+2 is less than
       or equal to c for any constant c and all n,n>=n0.
       ");.............// O(1) means computing time is
       constant.
14  printf ("\n \n 10n^2+4n+2 is not O(n)");
```

Scilab code Exa 1.13 `example`

```
1  clear ;
2  clc ;
3  printf ("\n Example 1.13");
```

13

```
4  printf(" \n \n [Omega] f(n)=omega(g(n))");
5  printf(" \n \n 3n+2=omega(n) as 3n+2>=3n for n>=1");
6  printf(" \n \n 3n+3=omega(n) as 3n+3>=3n for n>=1");
7  printf(" \n \n 100n+6=omega(n) as 100n+6>=100n for n
       >=1");
8  printf(" \n \n 10n^2+4n+2=omega(n^2) as 10n^2+4n+2>=n
       ^2 for n>=1");
9  printf(" \n \n 6*2^n+n^2=omega(n) as 6*2^n+n^2>=2^n
       for n>=1");
10 printf(" \n \n 3n+3=omega(1) ");
11 printf(" \n \n \t [Omega] f(n)=omega(1)");
```

Scilab code Exa 1.14 `example`

```
1  clear;
2  clc;
3  printf(" \n Example 1.14");
4  printf(" \n \n [Theta] f(n)=theta(g(n))");
5  printf(" \n \n 3n+2=theta(n) as 3n+2>=3n for al  n>=2
       ");
6  printf(" \n \n 3n+3=theta(n)");
7  printf(" \n \n 10n^2+4n+2=theta(n^2)");
8  printf(" \n \n 6*2^n+n^2=theta(2^n)");
9  printf(" \n \n 3n+2 is not theta(1) ");
10 printf(" \n \n 3n+3 is not theta(n^2) \n");
11 printf(" \n \n The Theta notation is more precise
       than both big oh and omega notaion");
```

Scilab code Exa 1.15 `example`

```
1  clear;
2  clc;
3  printf(" \n \t Example 1.15");
```

```scilab
 4  // how various functions grow with n, plotting of
        various functions is being shown.
 5  // like function 2^n grows very rapidly with n. and
        utility of programs with exponential complexity
        is limited to small n ( typically n<=40).
 6  n=[ 1 2 3 4 5 6];.......// takin value of n from 1
        to 10 to observe the variation in various
        functions.
 7  plot(log (n));
 8  plot(2^n);
 9  plot(n);
10  plot(n^2);
11  xtitle("Plot of function values","n -->","f -->");
12  printf(" \n \n X - axis is represented by values of
        n and Y-axis if represented by f");
```

Chapter 2

Arrays and Structures

Scilab code Exa 2.1 example

```
1 clear ;
2 clc ;
3 printf ("\n example 2.1") ;
4 // printing out values of the array .
5 a=[31 40 57 46 97 84];
6 printf ("\nvalues are :\n") ;
7 disp (a) ;
```

Scilab code Exa 2.2 example

```
1 clear ;
2 clc ;
3 printf ("\n Example 2.2\n") ;
4 // String insertion .
5 s="auto";...............//1st string or character
      array .
6 x="mobile";...............//2nd string or character
      array .
```

```
7  z=s+x;..........//concatenation of 2 strings.
8  printf("\tstring s=");
9  disp(s);
10 printf("\tstring x=");
11 disp(x);
12 printf("\tconcatenated string z=");
13 disp(z);........//dispalying concatenated string.
```

Scilab code Exa 2.3 example

```
1  clear;
2  clc;
3  printf("\nExample 2.3\n");
4  // comparision of 2 strings.
5  a="hakunah";..........//string 1.
6  b="matata";.............//string 2.
7  disp(" a & b respectively are =");
8  disp(a);
9  disp(b);
10 disp("comparing strings");
11 z=strcmp(a,b);.........//comparision of 2 strings.
12 if(z==0)
13     printf("\nMATCHED\n");........// if strings
           matched strcmp returns 0.
14 else
15     printf("\nNOT MATCHED\n");..........// if string
           doesn't matched strcmp returns -1.
16     end
17      q="akash";
18      w="akash";
19      disp("q & w respectively are=");
20      disp(q);
21      disp(w);
22      disp("comparing strings");
23      x=strcmp(q,w);
```

```
24        if(x==0)
25      printf(" \nMATCHED\n") ;.......// if strings
            matched strcmp returns 0.
26 else
27      printf(" \nNOT MATCHED\n") ;.........// if string
            doesn't matched strcmp returns −1.
28      end
```

Chapter 3

Stacks and Queues

Scilab code Exa 1.1.b example

```
1  //Exercise  question  2:
2  //Implementing  Push  And  Pop  Functions:
3  function[y,sta1]=empty(sta)
4     y=0;
5     sta1=0;
6     if(sta.top==0)
7        y=0;
8     else
9        y=1;
10    end
11    sta1=sta
12  endfunction
13
14  function[sta]=push(stac,ele)
15     sta=0;
16     if(empty(stac)==0)
17        stac.a=ele;
18        stac.top=stac.top+1;
19     else
20        stac.a=[stac.a(:,:) ele]
21        stac.top=stac.top+1;
```

```
22      end
23      disp(stac);
24      sta=stac;
25      funcprot(0)
26  endfunction
27
28  function[ele,sta]=pop(stack)
29      ele='-1';
30      if(empty(stack)==0)
31        disp("Stack Underflow");
32        break;
33      else
34        ele=stack.a(stack.top);
35        stack.top=stack.top-1;
36        if(stack.top~=0)
37           b=stack.a(1);
38        for i2=2:stack.top
39           b=[b(:,:) stack.a(i2)];
40        end
41        stack.a=b;
42      else
43        stack.a='0';
44      end
45      end
46      disp(stack);
47      sta=stack;
48  endfunction
49  global stack
50  //Calling Routine:
51  stack=struct('a',0,'top',0);
52  stack=push(stack,4);
53  stack=push(stack,55);
54  stack=push(stack,199);
55  stack=push(stack,363);
56  [ele,stack]=pop(stack);
57  disp(stack,"After the above operations stack is:");
```

Scilab code Exa 3.1 example

```
1  clear;
2  clc;
3  printf("\nexample 3.1\n");
4  //stacks follow LIFO i.e last in first out. so
        printing out array from last to first will be
        same as stack.
5  a=[12;35;16;48;29;17;13]
6  i=7;
7  printf("\tstack =");
8  while i>0
9      printf("\n\t%d",a(i));
10     i=i-1;
11 end
```

Scilab code Exa 3.1.2 example

```
1  //Unolved Example 2:
2  clear;
3  clc;
4  disp("Unsolved example 2");
5  //Implementing Stack using union:
6  function[stack]=sta_union(etype,a)
7    stackelement=struct('etype',etype);
8    [k,l]=size(a);
9  select stackelement.etype,
10 case 'int' then
11   a=int32(a);
12   stack=struct('top',l,'items',a);,
13   case 'float' then
14   a=double(a);
```

21

```
15   stack=struct('top',l,'items',a);,
16   case 'char' then
17   a=string(a);
18   stack=struct('top',l,'items',a);,
19 end
20 disp(stack,"Stack is:");
21 endfunction
22 a=[32 12.34 232 32.322]
23 stack=sta_union('float',a)
24 stack=sta_union('int',a)
25 stack=sta_union('char',a)
```

Scilab code Exa 1.2.b example

```
1  //Unsolved Example 1
2  clear;
3  clc;
4  disp("example 3.7");
5  //To   determine the syntacticaly valid string
6  function[l]=strlen(x)
7    i=1;
8    l=0;
9    [j,k]=size(x)
10   for  i=1:k
11     l=l+length(x(i));
12   end
13 endfunction
14 function []=stringvalid(str)
15   str=string(str);
16   stack=struct('a','0','top',0);
17   l1=strlen(str);
18   valid=1;
19   l=1;
20   while(l<=l1)
21       if(str(l)=='('|str(l)=='['|str(l)=='{')
```

```matlab
22          if(stack.top==0)
23             stack.a=str(l);
24             stack.top=stack.top+1;
25          else
26             stack.a=[stack.a(:,:) str(l)];
27             stack.top=stack.top+1;
28          end
29          disp(stack);
30        end
31        if(str(l)==')'|str(l)==']'|str(l)=='}')
32           if(stack.top==0)
33              valid=0;
34              break;
35           else
36              i=stack.a(stack.top);
37              b=stack.a(1);
38              for i1=2:stack.top-1
39                 b=[b(:,:) stack.a(i1)]
40              end
41              stack.a=b;
42             stack.top=stack.top-1;
43             symb=str(l);
44             disp(stack);
45             if(((symb==')')&(i=='('))|((symb==']')&(i==
                  '['))|((symb=='}')&(i=='{')))
46           else
47              valid=0;
48              break;
49           end
50        end
51      end
52        l=l+1;
53      end
54      if(stack.top~=0)
55         valid=0;
56      end
57      if(valid==0)
58         disp("Invalid String");
```

```
59        else
60          disp("Valid String");
61        end
62     endfunction
63     //Calling Routine:
64     stringvalid(['(' 'A' '+' 'B' '}' ')'])
65     stringvalid(['{' '[' 'A' '+' 'B' ']'  '-' '[' '('
           'C' '-'  'D' ')' ']'])
66     stringvalid(['(' 'A' '+' 'B' ')' '-' '{' 'C' '+' '
           D' '}' '-' '[' 'F' '+' 'G' ']'])
67     stringvalid(['(' '(' 'H' ')' '*' '{' '(' '[' 'J' '
           +' 'K' ']' ')' '}' ')'])
68     stringvalid(['(' '(' '(' 'A' ')' ')' ')'])
```

Scilab code Exa 3.2 example

```
1  //example 3.2
2  //Queue  Operations
3  clear;
4  clc;
5  function[q2]=push(ele,q1)
6    if(q1.rear==q1.front)
7      q1.a=ele;
8      q1.rear=q1.rear+1;
9    else
10     q1.a=[q1.a(:,:) ele];
11     q1.rear=q1.rear+1;
12   end
13   q2=q1;
14 endfunction
15 funcprot(0);
16 function[ele,q2]=pop(q1)
17   ele=-1;
18   q2=0;
19   if(q1.rear==q1.front)
```

24

```
20      disp("Queue Underflow");
21      return;
22    else
23      ele=q1.a(q1.rear-q1.front);
24      q1.front=q1.front+1;
25      i=1;
26      a=q1.a(1);
27      for i=2:(q1.rear-q1.front)
28         a=[a(:,:) q1.a(i)];
29      end
30      q1.a=a;
31    end
32    q2=q1;
33 endfunction
34 funcprot(0);
35 //Calling Routine:
36 q1=struct('a',0,'rear',0,'front',0)
37 q1=push(3,q1)
38 q1=push(22,q1);
39 q1=push(21,q1);
40 disp(q1,"Queue after insertion");
41 [ele,q1]=pop(q1)
42 disp(ele,"poped element");
43 disp(q1,"Queue after poping");
44 [ele,q1]=pop(q1);
45 [ele,q1]=pop(q1);
46 [ele,q1]=pop(q1);//Underflow Condition
```

Scilab code Exa 1.3.a example

```
1 clear;
2 clc;
3 disp("Unsolved example 3");
4 function[l]=strlen(x)
5    i=1;
```

```
 6    l=0;
 7    [j,k]=size(x)
 8    for   i=1:k
 9      l=l+length(x(i));
10    end
11  endfunction
12  function[]=str(st)
13    stack=struct('a',0,'top',0);
14    st=string(st);
15    l=1;
16    l1=strlen(st);
17    symb=st(l);
18    valid=1;
19    while(l<l1)
20      while(symb~='C')
21        if(stack.top==0)
22            stack.a=st(l);
23            stack.top=stack.top+1;
24          else
25            stack.a=[stack.a(:,:) st(l)];
26            stack.top=stack.top+1;
27          end
28          l=l+1;
29          symb=st(l);
30        end
31        i=st(l+1);
32      if(stack.top==0)
33          valid=0;
34          break;
35        else
36          symb1=stack.a(stack.top);
37          stack.top=stack.top-1;
38          if(i~=symb1)
39            valid=0;
40            break;
41          end
42        end
43      l=l+1;
```

```
44     end
45       if(stack.top~=0)
46          valid=0;
47       end
48       if(valid==0)
49          disp("Not of the given format");
50       else
51          disp("String Of the Given Format");
52       end
53  endfunction
54  //Calling Routine:
55  st=['A' 'A' 'B' 'A' 'C' 'A' 'B' 'A' 'A']
56  str(st)
57  st=['A' 'A' 'B' 'A' 'C' 'A' 'B' 'A' ]
58  str(st)
```

Scilab code Exa 3.3 example

```
1  //Solved Example 3.3:
2  //Convering an infix expression to a Postfix
       Expression:
3  function [sta]=push(stac,ele)
4     sta=0;
5     if(stac.top==0)
6       stac.a=ele;
7       stac.top=stac.top+1;
8     else
9       stac.a=[stac.a(:,:) ele]
10      stac.top=stac.top+1;
11    end
12    disp(stac);
13    sta=stac;
14  endfunction
15
16  function [ele,sta]=pop(stack)
```

```
17    ele='-1';
18    if(stack.top==0)
19      disp("Stack Underflow");
20      break;
21    else
22      ele=stack.a(stack.top);
23      stack.top=stack.top-1;
24      if(stack.top~=0)
25        b=stack.a(1);
26      for i2=2:stack.top
27        b=[b(:,:) stack.a(i2)];
28      end
29      stack.a=b;
30    else
31      stack.a='0';
32    end
33    end
34    sta=stack;
35  endfunction
36  function[l]=strlen(x)
37    i=1;
38    l=0;
39    [j,k]=size(x)
40    for  i=1:k
41      l=l+length(x(i));
42    end
43  endfunction
44  function[p]=pre(s1,s2)
45       i1=0;
46      select s1,
47      case '+' then i1=5;
48      case '-' then i1=5;
49      case '*' then  i1=9;
50      case '/' then i1=9;
51      end
52      i2=0;
53      select s2,
54      case '+' then i2=5;
```

```
55      case '-' then i2=5;
56      case '*' then  i2=9;
57      case '/' then i2=9;
58      end
59      p=0;
60      p=i1-i2;
61      if(s1=='(')
62         p=-1;
63      end
64      if(s2=='('&s1~=')')
65         p=-1;
66      end
67      if(s1~='('&s2==')')
68         p=1;
69      end
70
71    endfunction
72 function[a2]=intopo(a1,n)
73    stack=struct('a',0,'top',0);
74    l1=1;
75    l2=strlen(a1(1))
76    for i=2:n
77      l2=l2+strlen(a1(i))
78    end
79    a2=list();
80    while(l1<=l2)
81      symb=a1(l1);
82      if(isalphanum(string(a1(l1))))
83        a2=list(a2,symb);
84      else
85        while(stack.top~=0&(pre(stack.a(stack.top),
            symb)>=0))
86          [topsymb,stack]=pop(stack);
87          if(topsymb==')'|topsymb=='(')
88            a2=a2;
89          else
90            a2=list(a2,topsymb);
91          end
```

```
92          end
93          if ( stack . top ==0 | symb ~= ' ) ' )
94             stack = push ( stack , symb );
95          else
96             [ ele , stack ]= pop ( stack );
97          end
98        end
99        l1 = l1 +1;
100     end
101     while ( stack . top ~=0)
102        [ topsymb , stack ]= pop ( stack );
103        if ( topsymb == ' ) ' | topsymb == ' ( ' )
104           a2 = a2 ;
105        else
106           a2 = list ( a2 , topsymb );
107        end
108     end
109     disp ( a2 );
110  endfunction
111  // Calling Routine :
112  a1 =[ ' ( ' ' 2 ' ' + ' ' 3 ' ' ) ' ' * ' ' ( ' ' 5 ' ' - ' ' 4 ' ' ) ' ]
113  a2 = intopo ( a1 ,11)
```

Chapter 4

Linked lists

Scilab code Exa 4.1 example

```
1 //List of words in a linked list.
2 clear;
3 clc;
4 printf("\n Exapmle 4.1\n");
5 x=list('sci','lab','text','companionship','project')
    ;
6 disp("x=");
7 disp(x);
```

Scilab code Exa 4.2 example

```
1 //CIRCULAR LINKED LIST
2 clear;
3 clc;
4 funcprot(0);
5 disp("Example 4.2");
6 function [link2]=append(ele,link1)
7    link2=list
```

```
         (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, ,0,0)
          ;
 8    if(link1(1)(1).add==0)
 9      link1(1)(1).data=ele;
10      link1(1)(1).add=1;
11      link1(1)(1).nexadd=1;
12      link2(1)=link1(1)(1);
13      else
14      if(link1(1)(1).nexadd==link1(1)(1).add)
15        lin2=link1(1)(1);
16        lin2.data=ele;
17        lin2.add=link1(1)(1).add+1;
18        link1(1)(1).nexadd=lin2.add;
19        lin2.nexadd=link1(1)(1).add;
20        link2(1)=link1(1)(1);
21        link2(2)=lin2;
22      else
23        lin2=link1(1)(1);
24        i=1;
25        while(link1(i)(1).nexadd~=link1(1)(1).add)
26          i=i+1;
27        end
28        j=i;
29        lin2.data=ele;
30        lin2.add=link1(i).add+1;
31        lin2.nexadd=link1(1)(1).add;
32        link1(i).nexadd=lin2.add;
33        link2(1)=link1(1)(1);
34        i=2;
35        while(link1(i).nexadd~=lin2.add)
36          link2(i)=(link1(i));
37          i=i+1;
38        end
39        link2(i)=link1(i);
40        link2(i+1)=lin2;
41      end
42    end
43  endfunction
```

```
44  function[link2]=add(ele,pos,link1);
45     link2=list
          (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, ,0,0)
          ;
46     i=1;
47     while(i<=pos)
48       if(link1(i).nexadd==link1(1)(1).add)
49         break;
50       else
51          i=i+1;
52       end
53     end
54     if(link1(i).nexadd~=link1(1)(1).add)
55       i=i-1;
56        lin2.data=ele;
57        lin2.add=i;
58        j=i;
59        while(link1(j).nexadd~=link1(1)(1).add)
60           link1(j).add=link1(j).add+1;
61           link1(j).nexadd=link1(j).nexadd+1;
62           j=j+1;
63        end
64        link1(j).add=link1(j).add+1;
65        lin2.nexadd=link1(i).add;
66        link1(i-1).nexadd=lin2.add;
67        k=1;
68        while(k<i)
69           link2(k)=link1(k);
70           k=k+1;
71        end
72        link2(k)=lin2;
73        k=k+1;
74        link2(k)=link1(k-1);
75        k=k+1
76        l=k-1;
77        while(k~=j)
78           link2(k)=link1(l);
79           k=k+1;
```

```
80            l=l+1;
81          end
82          link2(j)=link1(j-1);;
83          link2(j+1)=link1(j);
84        else
85          if(i==pos)
86            k=1;
87            lin2.data=ele;
88            lin2.add=link1(i-1).add+1;
89            link1(i).add=link1(i).add+1;
90            lin2.nexadd=link1(i).add;
91            link1(i).nexadd=link1(1)(1).add;
92            k=1;
93            while(k<pos)
94              link2(k)=link1(k);
95              k=k+1;
96            end
97            link2(k)=lin2;
98            link2(k+1)=link1(k)
99          end
100       end
101
102 endfunction
103 function[link2]=delete1(pos,link1)
104   link2=list
        (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
        ;
105   i=1;
106   j=1;
107   while(i<pos)
108     if((link1(j).nexadd==link1(1)(1).add))
109       j=1;
110       i=i+1;
111     else
112       i=i+1;
113       j=j+1;
114     end
115   end
```

```
116     if(link1(j).nexadd~=link1(1)(1).add)
117       k=1;
118       if(j==1)
119         k=2;
120         while(link1(k).nexadd~=link1(1)(1).add)
121           link2(k-1)=link1(k);
122           k=k+1;
123         end
124         link2(k-1)=link1(k);
125         link2(k-1).nexadd=link2(1).add;
126       else
127         lin2=link1(j);
128         link1(j-1).nexadd=link1(j+1).add;
129         k=1;
130         while(link1(k).nexadd~=link1(j+1).add)
131           link2(k)=link1(k);
132           k=k+1;
133         end
134         link2(k)=link1(k);
135         k=k+2;
136         while(link1(k).nexadd~=link1(1)(1).add)
137           link2(k-1)=link1(k);
138           k=k+1;
139         end
140         link2(k-1)=link1(k);
141       end
142     else
143       link1(j-1).nexadd=link1(1)(1).add;
144       l=1;
145       while(link1(l).nexadd~=link1(1)(1).add)
146         link2(l)=link1(l);
147         l=l+1;
148       end
149       link2(l)=link1(l);
150     end
151 endfunction
152 //Calling Routine:
153 link1=struct('data',0,'add',0,'nexadd',0);
```

```
154  link1 = append (4 , link1 );// This will actualy create a
         list and 4 as start
155  link1 = append (6 , link1 );
156  link1 = add (10 ,2 , link1 );
157  link1 = delete1 (4 , link1 );// As the list is circular the
         4 'th element refers to actualy the 1 'st one
158  disp ( link1 ," After the above manuplations the list is
         ");
```

Scilab code Exa 4.3 example

```
1  // List Insertion .
2  clc ;
3  clear ;
4  disp (" Example 4.3");
5  funcprot (0)
6  function [ link2 ]= insert_pri ( ele , link1 )
7     link2 = list
          (0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , ,0 ,0)
          ;
8     if ( link1 (1) (1) . add ==0)
9        link1 (1) (1) . data = ele ;
10       link1 (1) (1) . add =1;
11       link1 (1) (1) . nexadd =1;
12       link2 (1)= link1 (1) (1) ;
13    else
14       if ( link1 (1) (1) . nexadd == link1 (1) (1) . add )
15          if ( ele >= link1 (1) (1) . data )
16             t = ele ;
17             p = link1 (1) (1) . data ;
18          else
19             t = link1 (1) (1) . data ;
20             p = ele ;
21          end
22          link1 (1) (1) . data = t ;
```

```
23      lin2=link1(1)(1);
24      lin2.data=p;
25      lin2.add=2;
26      lin2.nexadd=link1(1)(1).add;
27      link1(1)(1).nexadd=lin2.add;
28      link2(1)=link1(1)(1);
29      link2(2)=lin2;
30    else
31      i=1;
32      a=[];
33      while(link1(i).nexadd~=link1(1)(1).add)
34        a=[a(:,:) link1(i).data];
35        i=i+1;
36      end
37      a=[a(:,:) link1(i).data];
38      a=gsort(a);
39      j=1;
40      while(j<=i)
41        link1(j).data=a(j);
42        j=j+1;
43      end
44      k=1;
45      while(link1(k).data>=ele)
46        if(link1(k).nexadd==link1(1)(1).add)
47          break;
48        else
49          link2(k)=link1(k);
50          k=k+1;
51        end
52      end
53      if(link1(k).nexadd~=link1(1)(1).add)
54        lin2=link1(k);
55        lin2.data=ele;
56        lin2.add=link1(k).add;
57        j=k;
58        y=link1(1)(1).add;
59        while(link1(k).nexadd~=y)
60          link1(k).add=link1(k).add+1;
```

```
61            link1(k).nexadd=link1(k).nexadd+1;
62             k=k+1;
63           end
64          link1(k).add=link1(k).add+1;
65          lin2.nexadd=link1(j).add;
66          link2(j)=lin2;
67          j=j+1;
68          while(j<=k+1)
69            link2(j)=link1(j-1);
70            j=j+1;
71          end
72        else
73          lin2=link1(k);
74          lin2.data=ele;
75          lin2.nexadd=link1(1)(1).add;
76          lin2.add=link1(k).add+1;
77          link1(k).nexadd=lin2.add;
78          j=1;
79          while(j<=k)
80            link2(j)=link1(j);
81            j=j+1;
82          end
83          link2(j)=lin2;
84        end
85      end
86    end
87  endfunction
88  //Calling Routine:
89  link1=struct('data',0,'add',0,'nexadd',0);
90  link1=insert_pri(3,link1);
91  link1=insert_pri(4,link1);
92  link1=insert_pri(22,link1);
93  link1=insert_pri(21,link1);
94  link1=insert_pri(11,link1);
95  disp(link1,"List After Insertions");
```

Scilab code Exa 4.4 example

```
1  // Deletion  from  the  list :
2  function [ link2 ]= append ( ele , link1 )
3     link2 = list
          (0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 , ,0 ,0)
          ;
4     if ( link1 (1) (1) . add ==0)
5        link1 (1) (1) . data = ele ;
6        link1 (1) (1) . add =1;
7        link1 (1) (1) . nexadd =0;
8        link1 (1) (1) . prevadd =0;
9        link2 (1) = link1 (1) (1) ;
10       else
11       if ( link1 (1) (1) . nexadd ==0)
12          lin2 = link1 (1) (1) ;
13          lin2 . data = ele ;
14          lin2 . add = link1 (1) (1) . add +1;
15          link1 (1) (1) . nexadd = lin2 . add ;
16          lin2 . nexadd =0;
17          lin2 . prevadd = link1 (1) (1) . add ;
18          link2 (1) = link1 (1) (1) ;
19          link2 (2) = lin2 ;
20       else
21          lin2 = link1 (1) (1) ;
22          i =1;
23          while ( link1 ( i ) (1) . nexadd ~=0)
24             i = i +1;
25          end
26          j = i ;
27          lin2 . data = ele ;
28          lin2 . add = link1 ( i ) . add +1;
29          lin2 . nexadd =0;
30          link1 ( i ) . nexadd = lin2 . add ;
```

39

```
31          lin2.prevadd=link1(i).add;
32          link2(1)=link1(1)(1);
33          i=2;
34          while(link1(i).nexadd~=lin2.add)
35            link2(i)=(link1(i));
36            i=i+1;
37          end
38          link2(i)=link1(i);
39          link2(i+1)=lin2;
40        end
41      end
42  endfunction
43  function[link2]=add(ele,pos,link1);
44      link2=list
            (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
            ;
45      i=1;
46      while(i<=pos)
47        if(link1(i).nexadd==0)
48          break;
49        else
50          i=i+1;
51        end
52      end
53      if(link1(i).nexadd~=0)
54        i=i-1;
55          lin2.data=ele;
56          lin2.add=i;
57          j=i;
58          while(link1(j).nexadd~=0)
59            link1(j).prevadd=link1(j).prevadd+1;
60            link1(j).add=link1(j).add+1;
61            link1(j).nexadd=link1(j).nexadd+1;
62            j=j+1;
63          end
64          link1(j).prevadd=link1(j).prevadd+1;
65          link1(j).add=link1(j).add+1;
66          lin2.nexadd=link1(i).add;
```

```
67          link1(i).prevadd=lin2.add;
68          lin2.prevadd=link1(i-1).add;
69          link1(i-1).nexadd=lin2.add;
70          k=1;
71          while(k<i)
72              link2(k)=link1(k);
73              k=k+1;
74           end
75           link2(k)=lin2;
76           k=k+1;
77           link2(k)=link1(k-1);
78           k=k+1
79           l=k-1;
80           while(k~=j)
81              link2(k)=link1(l);
82              k=k+1;
83              l=l+1;
84           end
85           link2(j)=link1(j-1);;
86           link2(j+1)=link1(j);
87         else
88           if(i==pos)
89             k=1;
90             lin2.data=ele;
91             lin2.add=link1(i-1).add+1;
92             link1(i).add=link1(i).add+1;
93             lin2.nexadd=link1(i).add;
94             link1(i).prevadd=lin2.add;
95             lin2.prevadd=link1(i-1).add;
96             k=1;
97             while(k<pos)
98                link2(k)=link1(k);
99                k=k+1;
100            end
101            link2(k)=lin2;
102            link2(k+1)=link1(k)
103          end
104        end
```

```scilab
105
106  endfunction
107  function[link2]=delete1(pos,link1)
108    link2=list
         (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, ,0,0)
         ;
109    i=1;
110    while(i<=pos)
111      if((link1(i).nexadd==0))
112        break;
113      else
114        i=i+1;
115      end
116    end
117    if(link1(i).nexadd~=0)
118      i=i-1;
119      j=1;
120      if(i==1)
121        j=1;
122        while(link1(j).nexadd~=0)
123          link2(j)=link1(j);
124          j=j+1;
125        end
126        link2(j)=link1(j);
127      else
128        link1(i-1).nexadd=link1(i+1).add;
129        link1(i+1).prevadd=link1(i-1).add;
130      while(link1(j).nexadd~=link1(i+1).add)
131        link2(j)=link1(j);
132        j=j+1;
133      end
134      if(j~=i-1)
135        link2(j)=link1(j);
136        link2(j+1)=link1(j+1);
137        k=i+1;
138        l=2;
139      else
140        link2(j)=link1(j);
```

```
141        k=i+1;
142        l=1;
143      end
144      while(link1(k).nexadd~=0)
145        link2(j+l)=link1(k);
146        k=k+1;
147        l=l+1;
148      end
149      link2(j+l)=link1(k);
150    end
151    else
152      if(i==pos)
153        j=1;
154        link1(i-1).nexadd=0;
155        while(j<=i-1)
156          link2(j)=link1(j);
157          j=j+1;
158        end
159      end
160    end
161 endfunction
162 //Calling Routine:
163 link1=struct('data',0,'add',0,'nexadd',0);
164 link1=append(4,link1);
165 link1=append(6,link1);
166 link1=add(10,2,link1);
167 link1=delete1(3,link1);
168 disp(link1,"After the above manipulation the list is
      ");
```

Chapter 5

Trees

Scilab code Exa 5.1 example

```
1
2  funcprot (0) ;
3  function [ tree ]= maketree ( x )
4     tree = zeros (30 ,1) ;
5     for  i =1:30
6        tree ( i ) = -1;
7     end
8     tree (1) = x ;
9     tree (2) = -2;
10 endfunction
11 function [ tree1 ]= setleft ( tree , tre , x )
12    tree1 =[] ;
13    i =1;
14    while ( tree ( i ) ~= -2)
15       if ( tree ( i ) == tre )
16          j = i ;
17       end
18       i = i +1;
19    end
20    if ( i >2* j )
21       tree (2* j ) = x ;
```

```
22    else
23       tree(2*j)=x;
24       tree(2*j+1)=-2;
25       for l=i:2*j-1
26          tree(i)=-1;
27       end
28    end
29    tree1=tree;
30 endfunction
31 function[tree1]=setright(tree,tre,x)
32    tree1=[];
33    i=1;
34    while(tree(i)~=-2)
35       if(tree(i)==tre)
36          j=i;
37       end
38       i=i+1;
39    end
40    if(i>2*j+1)
41       tree(2*j+1)=x;
42    else
43       tree(2*j+1)=x;
44       tree(2*j+2)=-2;
45       for l=i:2*j
46          tree(i)=-1;
47       end
48    end
49    tree1=tree;
50 endfunction
51 function[x]=isleft(tree,tre)
52    i=1;
53    x=0;
54    while(tree(i)~=-2)
55       if(tree(i)==tre)
56          j=i;
57       end
58       i=i+1;
59    end
```

```scilab
60    if(i>=2*j)
61      if((tree(2*j)~=-1)|(tree(2*j)~=-2))
62        x=1;
63        return 1;
64      else
65        return 0;
66      end
67    else
68      x=0;
69      return x;
70    end
71  endfunction
72  function [x]=isright(tree,tre)
73    i=1;
74    x=0;
75    while(tree(i)~=-2)
76      if(tree(i)==tre)
77        j=i;
78      end
79      i=i+1;
80    end
81    if(i>=2*j+1)
82      if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
83        x=1;
84        return 1;
85      else
86        return 0;
87      end
88    else
89      x=0;
90      return x;
91    end
92  endfunction
93  //Calling Routine:
94  tree=maketree(3);
95  disp(tree,"Tree made");
96  tree=setleft(tree,3,1);
97  disp(tree,"After setting 1 to left of 3");
```

```
 98  tree=setright(tree,3,2);
 99  disp(tree,"After setting 2 to right of 3");
100  tree=setright(tree,2,4);
101  tree=setleft(tree,2,5);
102  tree=setright(tree,1,6);
103  tree=setright(tree,5,8);
104  disp(tree,"After above operations:");
105  x=isright(tree,3);
106  disp(x,"Checking for the right son of 3 yes if 1
        else no");
107  x=isleft(tree,2);
108  disp(x,"Check for left son of 2");
```

Scilab code Exa 5.2 example

```
 1  funcprot(0);
 2  function[tree]=maketree(x)
 3    tree=zeros(30,1);
 4    for i=1:30
 5      tree(i)=-1;
 6    end
 7    tree(1)=x;
 8    tree(2)=-2;
 9  endfunction
10  function[tree1]=setleft(tree,tre,x)
11    tree1=[];
12    i=1;
13    while(tree(i)~=-2)
14      if(tree(i)==tre)
15        j=i;
16      end
17      i=i+1;
18    end
19    if(i>2*j)
20      tree(2*j)=x;
```

47

```
21     else
22        tree(2*j)=x;
23        tree(2*j+1)=-2;
24        for  l=i:2*j-1
25           tree(i)=-1;
26        end
27     end
28     tree1=tree;
29  endfunction
30  function[tree1]=setright(tree,tre,x)
31     tree1=[];
32     i=1;
33     while(tree(i)~=-2)
34        if(tree(i)==tre)
35           j=i;
36        end
37        i=i+1;
38     end
39     if(i>2*j+1)
40        tree(2*j+1)=x;
41     else
42        tree(2*j+1)=x;
43        tree(2*j+2)=-2;
44        for  l=i:2*j
45           tree(i)=-1;
46        end
47     end
48     tree1=tree;
49  endfunction
50  function[x]=isleft(tree,tre)
51     i=1;
52     x=0;
53     while(tree(i)~=-2)
54        if(tree(i)==tre)
55           j=i;
56        end
57        i=i+1;
58     end
```

```
59    if(i>=2*j)
60      if((tree(2*j)~=-1)|(tree(2*j)~=-2))
61        x=1;
62        return 1;
63      else
64        return 0;
65      end
66    else
67      x=0;
68      return x;
69    end
70  endfunction
71  function[x]=isright(tree,tre)
72    i=1;
73    x=0;
74    while(tree(i)~=-2)
75      if(tree(i)==tre)
76        j=i;
77      end
78      i=i+1;
79    end
80    if(i>=2*j+1)
81      if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
82        x=1;
83        return 1;
84      else
85        return 0;
86      end
87    else
88      x=0;
89      return x;
90    end
91  endfunction
92  funcprot(0);
93  function[]=inorder(tree,p)
94    if(tree(p)==-1|tree(p)==-2)
95      return;
96    else
```

49

```scilab
 97        inorder(tree,2*p);
 98        printf("%d\t",tree(p));
 99        inorder(tree,2*p+1);
100     end
101  endfunction
102  function[]=preorder(tree,p)
103     if(tree(p)==-1|tree(p)==-2)
104        return;
105     else
106        printf("%d\t",tree(p));
107        preorder(tree,2*p);
108        preorder(tree,2*p+1);
109     end
110  endfunction
111  function[]=postorder(tree,p)
112     if(tree(p)==-1|tree(p)==-2)
113        return;
114     else
115        postorder(tree,2*p);
116        postorder(tree,2*p+1);
117        printf("%d\t",tree(p));
118     end
119  endfunction
120  //Calling Routine:
121  tree=maketree(3);
122  tree=setleft(tree,3,1);
123  tree=setright(tree,3,2);
124  tree=setleft(tree,2,4);
125  tree=setright(tree,2,5);
126  disp("Inorder traversal");
127  inorder(tree,1);
128  disp("Preorder traversal");
129  preorder(tree,1);
130  disp("Postorder traversal");
131  postorder(tree,1);
```

Scilab code Exa 5.3 example

```
1  funcprot(0);
2  function[tree]=maketree(x)
3     tree=zeros(1,30);
4     for i=1:30
5        tree(i)=-1;
6     end
7     tree(1)=x;
8     tree(2)=-2;
9  endfunction
10 function[tree1]=setleft(tree,tre,x)
11    tree1=[];
12    i=1;
13    while(tree(i)~=-2)
14       if(tree(i)==tre)
15          j=i;
16       end
17       i=i+1;
18    end
19    if(i>2*j)
20       tree(2*j)=x;
21    else
22       tree(2*j)=x;
23       tree(2*j+1)=-2;
24       for l=i:2*j-1
25          tree(i)=-1;
26       end
27    end
28    tree1=tree;
29 endfunction
30 function[tree1]=setright(tree,tre,x)
31    tree1=[];
32    i=1;
```

51

```
33    while(tree(i)~=-2)
34      if(tree(i)==tre)
35         j=i;
36      end
37      i=i+1;
38    end
39    if(i>2*j+1)
40      tree(2*j+1)=x;
41    else
42      tree(2*j+1)=x;
43      tree(2*j+2)=-2;
44      for l=i:2*j
45         tree(i)=-1;
46      end
47    end
48    tree1=tree;
49  endfunction
50  function[x]=isleft(tree,tre)
51    i=1;
52    x=0;
53    while(tree(i)~=-2)
54      if(tree(i)==tre)
55         j=i;
56      end
57      i=i+1;
58    end
59    if(i>=2*j)
60      if((tree(2*j)~=-1)|(tree(2*j)~=-2))
61         x=1;
62         return 1;
63      else
64         return 0;
65      end
66    else
67      x=0;
68      return x;
69    end
70  endfunction
```

```
71  function [x]=isright(tree,tre)
72    i=1;
73    x=0;
74    while(tree(i)~=-2)
75      if(tree(i)==tre)
76          j=i;
77      end
78      i=i+1;
79    end
80    if(i>=2*j+1)
81      if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
82        x=1;
83        return 1;
84      else
85        return 0;
86      end
87    else
88      x=0;
89      return x;
90    end
91  endfunction
92  funcprot(0);
93  function []=inorder(tree,p)
94    if(tree(p)==-1|tree(p)==-2)
95      return;
96    else
97      inorder(tree,2*p);
98      disp(tree(p),"   ");
99      inorder(tree,2*p+1);
100   end
101 endfunction
102 function []=preorder(tree,p)
103   if(tree(p)==-1|tree(p)==-2)
104     return;
105   else
106     disp(tree(p),"   ");
107     preorder(tree,2*p);
108     preorder(tree,2*p+1);
```

```scilab
109    end
110  endfunction
111  function []=postorder(tree,p)
112    if(tree(p)==-1|tree(p)==-2)
113      return;
114    else
115      postorder(tree,2*p);
116      postorder(tree,2*p+1);
117      disp(tree(p)," ");
118    end
119  endfunction
120  function [tree1]=binary(tree,x)
121    p=1;
122    while(tree(p)~=-1&tree(p)~=-2)
123      q=p;
124      if(tree(p)>x)
125        p=2*p;
126      else
127        p=2*p+1;
128      end
129    end
130    i=1;
131    while(tree(i)~=-2)
132      i=i+1;
133    end
134    if(tree(q)>x)
135      if(i==2*q)
136        tree(2*q)=x;
137        tree(2*q+1)=-2
138      else
139        if(i<2*q)
140          tree(i)=-1;
141          tree(2*q+1)=-2;
142          tree(2*q)=x;
143        end
144      end
145
146    else
```

```
147       if(i==2*q+1)
148          tree(2*q+1)=x;
149          tree(2*q+2)=-2;
150       else
151          if(i<2*q+1)
152             tree(i)=-1;
153             tree(2*q+1)=x;
154             tree(2*q+2)=-2;
155          end
156       end
157
158    end
159    tree1=tree;
160 endfunction
161 //Calling Routine:
162 tree=maketree(3);
163 tree=binary(tree,1);
164 tree=binary(tree,2);
165 tree=binary(tree,4);
166 tree=binary(tree,5);
167 disp(tree,"Binary tree thus obtaine by inserting
        1,2,4and5 in tree rooted 3 is:");
```

Scilab code Exa 5.4 example

```
1 function[tree1]=binary(tree,x)
2    p=1;
3    while(tree(p)~=-1&tree(p)~=-2)
4      q=p;
5      if(tree(p)>x)
6         p=2*p;
7      else
8         p=2*p+1;
9      end
10   end
```

```
11    if(tree(q)>x)
12      if(tree(2*q)==-2)
13        tree(2*q)=x;
14        tree(2*q+1)=-2;
15      else
16        tree(2*q)=x;
17      end
18    else
19      if(tree(2*q+1)==-2)
20        tree(2*q+1)=x;
21        tree(2*q+2)=-2;
22      else
23        tree(2*q+1)=x;
24      end
25    end
26    tree1=tree;
27  endfunction
28  funcprot(0);
29  function[tree]=maketree(x)
30    tree=zeros(40,1);
31    for i=1:40
32      tree(i)=-1;
33    end
34    tree(1)=x;
35    tree(2)=-2;
36  endfunction
37  function[]=duplicate1(a,n)
38    tree=maketree(a(1));
39    q=1;
40    p=1;
41    i=2;
42    x=a(i)
43    while(i<n)
44      while(tree(p)~=x&tree(q)~=-1&tree(q)~=-2)
45        p=q;
46        if(tree(p)<x)
47          q=2*p;
48        else
```

```
49          q=2*p+1;
50        end
51      end
52      if(tree(p)==x)
53        disp(x," Duplicate  ");
54      else
55        tree=binary(tree,x);
56      end
57      i=i+1;
58      x=a(i);
59    end
60    while(tree(p)~=x&tree(q)~=-1&tree(q)~=-2)
61        p=q;
62        if(tree(p)<x)
63          q=2*p;
64        else
65          q=2*p+1;
66        end
67      end
68      if(tree(p)==x)
69        disp(x," Duplicate  ");
70      else
71        tree=binary(tree,x);
72      end
73 endfunction
74 //Calling Adress:
75 a=[22 11 33 22 211 334]
76 duplicate1(a,6)
77 a=[21 11 33 22 22 334]
78 duplicate1(a,6)
```

Chapter 6

Graphs

Scilab code Exa 6.1 example

```
1  clear;
2  clc;
3  disp("Example 6.1");
4  //Depth First Search Traversal
5  funcprot(0)
6  function []=Dfs(adj,n);
7     i=1,j=1;
8     colour=[];
9     for i=1:n
10    for j=1:n
11        colour=[colour(:,:) 0];
12      end
13    end
14    disp("The DFS traversal is");
15  dfs(adj,colour,1,n);
16  endfunction
17  function []=dfs(adj,colour,r,n)
18    colour(r)=1;
19    disp(r,"   ");
20    for i=1:n
21      if(adj((r-1)*n+i)&(colour(i)==0))
```

```
22        dfs(adj,colour,i,n);
23      end
24    end
25    colour(r)=2;
26 endfunction
27 //Calling  Routine:
28 n=4;
29 adj=[0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0]
30 Dfs(adj,n)
```

Scilab code Exa 6.2 example

```
 1 clear;
 2 clc;
 3 disp("Example  6.2");
 4 ////BFS Traversal
 5 funcprot(0)
 6 function[q2]=push(ele,q1)
 7    if(q1.rear==q1.front)
 8      q1.a=ele;
 9      q1.rear=q1.rear+1;
10    else
11      q1.a=[q1.a(:,:) ele];
12      q1.rear=q1.rear+1;
13    end
14    q2=q1;
15 endfunction
16 function[ele,q2]=pop(q1)
17    ele=-1;
18    q2=0;
19    if(q1.rear==q1.front)
20          return;
21    else
22      ele=q1.a(q1.rear-q1.front);
23      q1.front=q1.front+1;
```

```scilab
24        i=1;
25        a=q1.a(1);
26        for i=2:(q1.rear-q1.front)
27            a=[a(:,:) q1.a(i)];
28        end
29        q1.a=a;
30    end
31    q2=q1;
32 endfunction
33
34 function []=Bfs(adj,n);
35    i=1,j=1;
36    colour=[];
37    for i=1:n
38    for j=1:n
39        colour=[colour(:,:) 0];
40      end
41    end
42    disp("The BFS Traversal is");
43 bfs(adj,colour,1,n);
44 endfunction
45 function []=bfs(adj,colour,s,n)
46    colour(s)=1;
47    q=struct('rear',0,'front',0,'a',0);
48    q=push(s,q);
49    while((q.rear)-(q.front)>0)
50      [u,q]=pop(q);
51      disp(u," ");
52      for i=1:n
53        if(adj((u-1)*n+i)&(colour(i)==0))
54           colour(i)=1;
55           q=push(i,q);
56        end
57      end
58      colour(u)=2;
59    end
60 endfunction
61 //Calling Routine:
```

60

```
62  n=4;
63  adj=[0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0]
64  Bfs(adj,n)
```

Scilab code Exa 6.3 example

```
1  clear;
2  clc;
3  disp("Example 6.3");
4  //Warshall's Algorithm
5  clc;
6  clear;
7  funcprot(0)
8  function[path]=transclose(adj,n)
9    for  i=1:n
10      for j=1:n
11        path((i-1)*n+j)=adj((i-1)*n+j);
12      end
13    end
14    for k=1:n
15      for  i=1:n
16        if(path((i-1)*n+k)==1)
17          for j=1:n
18            path((i-1)*n+j)=path((i-1)*n+j)|path((k-1)
                *n+j);
19          end
20        end
21      end
22    end
23    printf("Transitive closure for the given graph is
          :\n");
24    for  i=1:n
25      printf("For vertex %d \n",i);
26      for j=1:n
27        printf("%d %d is %d\n",i,j,path((i-1)*n+j));
```

```
28        end
29      end
30   endfunction
31   // Calling Routine :
32   n =3;
33   adj =[0 1 0 0 0 1 0 0 0]
34   path = transclose ( adj , n )
```

Scilab code Exa 6.4 example

```
1   clear ;
2   clc ;
3   disp ( " Example  6.4 " ) ;
4   // Finnding  Transitive  Closure
5   funcprot (0) ;
6   function [ path ]= Tranclose ( adj , n ) ;
7      i =1 , j =1;
8      path = zeros ( n * n ,1) ;
9      path = tranclose ( adj , n ) ;
10     printf ( " Transitive  Closure  Of  Given  Graph  is :\ n " ) ;
11     for  i =1: n
12       printf ( " For  Vertex  %d\ n " , i ) ;
13       for  j =1: n
14         printf ( "  %d %d  is  %d\ n " , i , j , path (( i -1) * n + j ) ) ;
15       end
16     end
17
18   endfunction
19   function [ path ]= tranclose ( adj , n )
20      adjprod = zeros ( n * n ,1) ;
21      k =1;
22      newprod = zeros ( n * n ,1) ;
23      for  i =1: n
24        for  j =1: n
25          path (( i -1) * n + j ) = adj (( i -1) * n + j ) ;
```

```
26          adjprod((i-1)*n+j)= path((i-1)*n+j);
27        end
28      end
29      for i=1:n
30        newprod=prod1(adjprod,adj,n);
31        for j=1:n
32          for k=1:n
33            path((j-1)*n+k)=path((j-1)*n+k)|newprod((j
                 -1)*n+k);
34          end
35        end
36        for j=1:n
37          for k=1:n
38            adjprod((j-1)*n+k)=newprod((j-1)*n+k);
39          end
40        end
41      end
42  endfunction
43  function[c]=prod1(a,b,n)
44      for i=1:n
45        for j=1:n
46          val=0
47          for k=1:n
48            val=val|(a((i-1)*n+k)&b((k-1)*n+j));
49          end
50          c((i-1)*n+j)=val;
51        end
52      end
53  endfunction
54  //Calling Routine:
55  n=3;
56  adj=[0 1 0 0 0 1 0 0 0]
57  path=Tranclose(adj,n)
```

Scilab code Exa 6.5 example

```
1  clear;
2  clc;
3  disp("Example 6.5");
4  //Finding The  Number Of Simple Paths From One Point
        To Another In A Given Graph
5  funcprot(0)
6  function []= sim_path(n,adj,i,j);
7     l=0;
8     m=1;
9     for m=1:n
10       l=l+path(m,n,adj,i,j);
11    end
12    printf("There are %d  simple  paths  from %d to %d
          in  the  given  graph\n",l,i,j);
13 endfunction
14 function [b]=path(k,n,adj,i,j)
15    b=0;
16    if(k==1)
17       b=adj((i-1)*n+j);
18    else
19       for c=1:n
20          if(adj((i-1)*n+c)==1)
21             b=b+path(k-1,n,adj,c,j);
22          end
23       end
24    end
25    return b;
26 endfunction
27 n=3;
28 adj=[0 1 1 0 0 1 0 0 0];
29 b=sim_path(n,adj,1,3)
```

Scilab code Exa 6.6 example

```
1  clear;
```

```scilab
2  clc;
3  disp("Example 6.6");
4  //Dijkstras Algorithm
5  funcprot(0)
6  function[l]=short(adj,w,i1,j1,n)
7    for i=1:n
8      for j=1:n
9        if(w((i-1)*n+j)==0)
10          w((i-1)*n+j)=9999;
11        end
12      end
13    end
14
15    distance=[];
16    perm=[];
17    for i=1:n
18      distance=[distance(:,:) 99999];
19      perm=[perm(:,:) 0];
20    end
21    perm(i1)=1;
22    distance(i1)=0;
23    current=i1;
24    while(current~=j1)
25      smalldist=9999;
26      dc=distance(current);
27      for i=1:n
28        if(perm(i)==0)
29          newdist=dc+w((current-1)*n+i);
30          if(newdist<distance(i))
31            distance(i)=newdist;
32          end
33          if(distance(i)<smalldist)
34            smalldist=distance(i);
35            k=i;
36          end
37        end
38      end
39      current=k;
```

```
40      perm ( current )=1;
41    end
42    l= distance ( j1 ) ;
43    printf (" The   shortest  path  between  %d and  %d is  %d
          " , i1 , j1 , l ) ;
44  endfunction
45  // Calling  Routine :
46  n =3;
47  adj =[0  1  1  0  0  1  0  0  0]// Adjacency   List
48  w=[0  12  22  0  0  9  0  0  0]// weight  list  fill  0  for  no
        edge
49  short ( adj , w , 1 , 3 , n ) ;
```

Scilab code Exa 6.7 example

```
1  clear ;
2  clc ;
3  disp (" Example  6.7 ") ;
4  // Finding  The  Number  Of  Paths  From  One  Vertex  To
        Another  Of  A  Given  Length
5
6  function [b]= path ( k , n , adj , i , j )
7    b =0;
8    if ( k ==1)
9      b= adj (( i -1) * n+j ) ;
10   else
11     for  c =1: n
12       if ( adj (( i -1) * n+c ) ==1)
13         b=b+ path ( k -1 , n , adj , c , j ) ;
14       end
15     end
16   end
17     printf (" Number  of  paths  from  vertex  %d to  %d of
            length  %d are  %d" , i , j , k , b ) ;
18   return  b ;
```

66

```
19  endfunction
20  // Calling  Routine :
21  n =3;
22  adj =[0 1 1 0 0 1 0 0 0]
23  b= path (1 ,n , adj ,1 ,3)
```

Chapter 7

Sorting

Scilab code Exa 7.1 example

```
1  clear ;
2  clc ;
3  disp ("Example  7.1");
4  funcprot (0);
5  function [a1]= insertion (a,n)
6     for  k =1: n
7        y = a ( k ) ;
8        i = k ;
9        while ( i >=1)
10          if ( y < a ( i ) )
11             a ( i +1) = a ( i ) ;
12             a ( i ) = y ;
13          end
14          i = i -1;
15       end
16    end
17    a1 = a ;
18    disp ( a1 ," Sorted  array  is :") ;
19  endfunction
20  // Calling  Routine :
21  a =[5  4  3  2  1]          //  worst -case  behaviour  of
```

```
       insertion  sort .
22 disp (a ,"Given  Array") ;
23 a1=insertion (a ,5)
```

Scilab code Exa 7.2 example

```
 1 clear ;
 2 clc ;
 3 disp ("Example  7.2") ;
 4 funcprot (0) ;
 5 function [a1]=insertion (a ,n)
 6    for  k=1:n
 7      y=a (k) ;
 8      i=k ;
 9      while (i >=1)
10        if (y <a (i))
11           a (i+1)=a (i) ;
12           a (i)=y ;
13        end
14        i=i -1;
15      end
16    end
17    a1=a ;
18    disp (a1 ,"Sorted  array  is :") ;
19 endfunction
20 // Calling  Routine :
21 a=[2  3  4  5  1]
22 disp (a ,"Given  Array") ;
23 a1=insertion (a ,5)
```

Scilab code Exa 7.3 example

```
 1 clear ;
```

```
2 clc;
3 disp("Example 7.3");
4 funcprot(0);
5 function[a1]=quick(a);
6   a=gsort(a);//IN BUILT QUICK SORT FUNCTION
7   n=length(a);
8   a1=[];
9   for i=1:n
10     a1=[a1(:,:) a(n+1-i)];
11   end
12   disp(a1,"Sorted array is:");
13 endfunction
14 //Calling Routine:
15 a=[26 5 37 1 61 11 59 15 48 19]
16 disp(a,"Given Array");
17 a1=quick(a)
```

Scilab code Exa 7.4 example

```
1 clear;
2 clc;
3 disp("Example 7.4");
4 function[a1]=insertion(a,n)
5   for k=1:n
6     y=a(k);
7     i=k;
8     while(i>=1)
9       if(y<a(i))
10        a(i+1)=a(i);
11        a(i)=y;
12       end
13       i=i-1;
14     end
15   end
16   a1=a;
```

70

```
17    disp(a1,"Sorted  array  is:");
18 endfunction
19 //Calling  Routine:
20 a=[3 1 2]
21 disp(a,"Given  Array");
22 a1=insertion(a,3)
```

Scilab code Exa 7.5 example

```
1 clear;
2 clc;
3 disp("Example  7.5");
4 funcprot(0);
5 function[a1]=mergesort(a,p,r)
6    if(p<r)
7      q=int((p+r)/2);
8      a=mergesort(a,p,q);
9      a=mergesort(a,q+1,r);
10      a=merge(a,p,q,r);
11    else
12      a1=a;
13      return;
14    end
15    a1=a;
16 endfunction
17 function[a1]=merge(a,p,q,r)
18    n1=q-p+1;
19    n2=r-q;
20    left=zeros(n1+1);
21    right=zeros(n2+1);
22    for i=1:n1
23      left(i)=a(p+i-1);
24    end
25    for i1=1:n2
26        right(i1)=a(q+i1);
```

```
27    end
28    left(n1+1)=999999999;
29    right(n2+1)=999999999;
30    i=1;
31    j=1;
32    k=p;
33    for  k=p:r
34      if(left(i)<=right(j))
35        a(k)=left(i);
36        i=i+1;
37      else
38        a(k)=right(j);
39        j=j+1;
40      end
41    end
42    a1=a;
43  endfunction
44  // Calling  Routine:
45  a=[26  5  77  1  61  11  59  15  48  19]
46  disp(a,"Given  Array");
47  a1=mergesort(a,1,10)
48  disp(a1,"Sorted  array  is:");
```

Scilab code Exa 7.6 example

```
1  clear;
2  clc;
3  disp("Example  7.7");
4  function[a1]=shell(a,n,incr,nic)
5    for i=1:nic
6      span=incr(i);
7      for j=span+1:n
8        y=a(j);
9        k=j-span;
10       while(k>=1&y<a(k))
```

```
11              a(k+span)=a(k);
12           k=k-span;
13         end
14         a(k+span)=y;
15       end
16   end
17   a1=a;
18   disp(a1,"Sorted array is:");
19 endfunction
20 //Calling Routine:
21 a=[23 21 232 121 2324 1222433 1212]
22 disp(a,"Given Array");
23 incr=[5 3 1]//must always end with 1
24 a1=shell(a,7,incr,3)
```

Scilab code Exa 7.7 example

```
1 clear;
2 clc;
3 disp("Example 7.7");
4 function[a1]=shell(a,n,incr,nic)
5   for i=1:nic
6     span=incr(i);
7     for j=span+1:n
8         y=a(j);
9         k=j-span;
10        while(k>=1&y<a(k))
11             a(k+span)=a(k);
12           k=k-span;
13         end
14         a(k+span)=y;
15       end
16   end
17   a1=a;
18   disp(a1,"Sorted array is:");
```

```
19  endfunction
20  // Calling  Routine :
21  a =[23  21  232  121  2324  1222433  1212]
22  disp (a ," Given  Array ") ;
23  incr =[5  3  1]// must  always  end  with  1
24  a1 = shell (a ,7 , incr ,3)
```

Scilab code Exa 7.8 example

```
1  clear ;
2  clc ;
3  function []= sortedsearch (a ,n , ele )
4    if (a (1) > ele | a (n) < ele )
5      disp (" NOT  IN  THE  LIST ") ;
6    else
7      i =1;
8      j =0;
9      for  i =1: n
10       if (a (i) == ele )
11         printf (" FOUND  %d  AT  %d", ele ,i );
12         j =1;
13       else
14         if (a (i) > ele )
15            break ;
16         end
17       end
18     end
19     if (j ==0)
20       disp ("%d  NOT  FOUND ", ele );
21     end
22   end
23  endfunction
24  // Calling  Routine :
25  a =[2  22  23  33  121  222  233]// a  should  be   sorted
26  disp (a ," Given  array ") ;
```

```
27  sortedsearch(a,7,23)
```

```
27  sortedsearch(a,7,23)
```

Chapter 8

Hashing

Scilab code Exa 8.1 example

```
1  clear ;
2  clc ;
3  disp ( ” Example  8 . 1 ” ) ;
4  k =12320324111220;
5  p1 =123;
6  p2 =203;
7  p3 =241;......//key  k  partitioned  into  parts  that  are
          3  decimal  long .
8  p4 =112;
9  p5 =20;
10  // /..... using  shift  folding ...
11  // ..... partitions  are  added  to  get  the  hash  address .
12  z=p1+p2+p3+p4+p5;
13  disp (z) ;
14  //when  folding  at  the  boundaries  is  used ,we  reverse
       p2  and  p4 .
15   p2 =302;
16   p4 =211;
17   x=p1+p2+p3+p4+p5;
18   disp (x) ;
```

Scilab code Exa 8.2 example

```
1  clear;
2  clc;
3  disp("Example 8.2");
4  function []= stringtoint()
5      num= ascii("scilab");
6      disp("displayin ascii codes of alphabets=");
7      disp(num);
8      // converting strings into unique non-negative
           integer and suming these unique integers.
9      z=sum(num);
10     disp("displayin sum of these integers");
11     disp(z);
12  endfunction
13  stringtoint()
```

Chapter 9

Priority Queues

Scilab code Exa 9.1 example

```
1  clear;
2  clc;
3  //Implementing Priority Queue Using Lists
4  funcprot(0)
5  function[link2]=insert_pri(ele,link1)
6    link2=list
         (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
         ;
7    if(link1(1)(1).add==0)
8      link1(1)(1).data=ele;
9      link1(1)(1).add=1;
10     link1(1)(1).nexadd=1;
11     link2(1)=link1(1)(1);
12   else
13     if(link1(1)(1).nexadd==link1(1)(1).add)
14       if(ele>=link1(1)(1).data)
15         t=ele;
16         p=link1(1)(1).data;
17       else
18         t=link1(1)(1).data;
19         p=ele;
```

```
20          end
21          link1(1)(1).data=t;
22          lin2=link1(1)(1);
23          lin2.data=p;
24          lin2.add=2;
25          lin2.nexadd=link1(1)(1).add;
26          link1(1)(1).nexadd=lin2.add;
27          link2(1)=link1(1)(1);
28          link2(2)=lin2;
29        else
30          i=1;
31          a=[];
32          while(link1(i).nexadd~=link1(1)(1).add)
33            a=[a(:,:) link1(i).data];
34            i=i+1;
35          end
36          a=[a(:,:) link1(i).data];
37          a=gsort(a);
38          j=1;
39          while(j<=i)
40            link1(j).data=a(j);
41            j=j+1;
42          end
43          k=1;
44          while(link1(k).data>=ele)
45            if(link1(k).nexadd==link1(1)(1).add)
46              break;
47            else
48              link2(k)=link1(k);
49              k=k+1;
50            end
51          end
52          if(link1(k).nexadd~=link1(1)(1).add)
53            lin2=link1(k);
54            lin2.data=ele;
55            lin2.add=link1(k).add;
56            j=k;
57            y=link1(1)(1).add;
```

79

```
58              while(link1(k).nexadd~=y)
59                 link1(k).add=link1(k).add+1;
60                 link1(k).nexadd=link1(k).nexadd+1;
61                 k=k+1;
62              end
63              link1(k).add=link1(k).add+1;
64              lin2.nexadd=link1(j).add;
65              link2(j)=lin2;
66              j=j+1;
67              while(j<=k+1)
68                 link2(j)=link1(j-1);
69                 j=j+1;
70              end
71           else
72              lin2=link1(k);
73              lin2.data=ele;
74              lin2.nexadd=link1(1)(1).add;
75              lin2.add=link1(k).add+1;
76              link1(k).nexadd=lin2.add;
77              j=1;
78              while(j<=k)
79                 link2(j)=link1(j);
80                 j=j+1;
81              end
82              link2(j)=lin2;
83           end
84        end
85     end
86   endfunction
87   function[ele,link2]=extract_min(link1);
88     link2=list
          (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0,0)
          ;
89     i=1;
90     ele=-1;
91     if(link1(1)(1).add==0)
92        disp("Underflow");
93        return;
```

```scilab
 94      else
 95        if(link1(1)(1).nexadd==link1(1)(1).add)
 96          link1(1)(1).add=0;
 97          link1(1)(1).nexadd=0;
 98          ele=link1(1)(1).data;
 99          link1(1)(1).data=0;
100          link2(1)=link1(1)(1);
101        else
102          i=1;
103          while(link1(i).nexadd~=link1(1)(1).add)
104             link2(i)=link1(i);
105             i=i+1;
106          end
107          ele=link1(i).data;
108          link2(i-1).nexadd=link2(1).add;
109        end
110      end
111    endfunction
112    //Calling Routine:
113    link1=struct('data',0,'add',0,'nexadd',0);
114    link1=insert_pri(3,link1);
115    link1=insert_pri(4,link1);
116    link1=insert_pri(22,link1);
117    link1=insert_pri(21,link1);
118    link1=insert_pri(11,link1);
119    disp(link1,"List After Insertions");
120    [ele,link1]=extract_min(link1)
121    disp(ele,"Element after the min extraction");
```