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

August 17, 2013

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Textbook Companion and Scilab codes written in it can be downloaded from the "Textbook Companion Project" section at the website http://scilab.in

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

```
//to do sorting of nos. contained in a list
function []=sorting(a)
    i=1;
    [j,k]=size(a);
    j=i;
    for i=1:k-1
        for j=i:k
            if a(i)>a(j) then
            z=a(i);
            a(i)=a(j);
            a(j)=z;
        end
    end
end
for i=1:k
    disp(a(i));
end

funcprot(0);
endfunction
    //callin routine
    a=[5 7 45 23 78]
    sort=sorting(a)
```

Scilab code Exa 1.2 example

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

Scilab code Exa 1.3 example

```
// to do binary search..
function []=search(a)
    i=1;
    [j,k]=size(a);
            for i=1:k
                if x==a(i) then
                    printf("\nFOUND and index no. is
                        =%d\t",i);
                end
            end
        funcprot(0);
endfunction
//callin routine
a=[5 7 45 28 99]
x=45
binary=search(a)
```

Scilab code Exa 1.4 example

```
// example 1.4
// permutation of a string or character array...
clear all;
clc;
x=['a' 'b' 'c' 'd']
printf("\npossible permutation of given string are\n
   ");
y=perms(x);
disp(y);
```

Scilab code Exa 1.5 example

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

```
        printf("\n",0);
        end
        if x==INT_MAX then      //
           NaturalNumberSuccessor(x)
            printf("\nans. is=%d",x);
        else
            printf("\nans. is=%d",x+1);
        end
    endfunction
    //callin routine
    x=56
    y=ADT(x);
```

Scilab code Exa 1.6 example

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

Scilab code Exa 1.7 example

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

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

Scilab code Exa 1.8 example

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

Scilab code Exa 1.9 example

```
clear all;
clc;
printf("\n Example 1.9");
```

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

Scilab code Exa 1.10 example

```
clear all;
clc;
printf("\n Example 1.10");
// function to sum a list of numbers.
function[]=add()
    printf("\n no. in the list are");
    disp(a);
    x=sum(a);
    printf("\n Result=%d",x);
    funcprot(0);
    endfunction
    //calling routine.
    a=[2 5 6 7 9 1 6 3 7 45]
    add()
```

Scilab code Exa 1.11 example

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

```
  .
```

Scilab code Exa 1.12 example

```
clear all;
clc;
printf("Example 1.12");
//[BIG "oh"]f(n)=O(g(n)). (big oh notation).
printf("\n \n 3n+2=O(n) as 3n+2<=4n for all n>=2.");
printf("\n \n 3n+3=O(n) as 3n+3<=4n for all n>=3.")
   ;.............// O(n) is called linear.
printf("\n \n 3n+2=O(n) as 100n+6<=101n for all n
   >=10.");
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.
printf("\n \n 1000n^2+100n-6=O(n^2) as 1000n^2+100n
   -6<=1001n^2 for n>=100.");
printf("\n \n 6*2^n+n^2<=7*2^n for n>=4");
printf("\n \n 3n+3=O(n^2) as 3n+3<=3n^2 for n>=2");
printf("\n \n 10n^2+4n+2=O(n^4) as 10n^2+4n+2<=10n^4
    for n>=2.");
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.
printf("\n \n 10n^2+4n+2 is not O(n)");
```

Scilab code Exa 1.13 example

```
clear all;
clc;
printf("\n Example 1.13");
printf("\n \n [Omega] f(n)=omega(g(n))");
printf("\n \n 3n+2=omega(n) as 3n+2>=3n for n>=1");
printf("\n \n 3n+3=omega(n) as 3n+3>=3n for n>=1");
printf("\n \n 100n+6=omega(n) as 100n+6>=100n for n
   >=1");
printf("\n \n 10n^2+4n+2=omega(n^2) as 10n^2+4n+2>=n
   ^2 for n>=1");
printf("\n \n 6*2^n+n^2=omega(n) as 6*2^n+n^2>=2^n
   for n>=1");
printf("\n \n 3n+3=omega(1) ");
printf(" \n \n \t [Omega] f(n)=omega(1)");
```

Scilab code Exa 1.14 example

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

Scilab code Exa 1.15 example

```
clear all;
clc;
printf("\n \t Example 1.15");
```

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

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

Scilab code Exa 2.2 example

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

```
z=s+x;..........//concatenation of 2 strings.
printf("\tstring s=");
disp(s);
printf("\tstring x=");
disp(x);
printf("\tconcatenated string z=");
disp(z);........//dispalying concatenated string.
```

Scilab code Exa 2.3 example

```
clear all;
clc;
printf("\nExample 2.3\n");
// comparision of 2 strings.
a="hakunah";..........//string 1.
b="matata";.............//string 2.
disp(" a & b respectively are =");
disp(a);
disp(b);
disp("comparing strings");
z=strcmp(a,b);.........//comparision of 2 strings.
if(z==0)
    printf("\nMATCHED\n");.......// if strings
       matched strcmp returns 0.
else
    printf("\nNOT MATCHED\n");.........// if string
       doesn't matched strcmp returns -1.
    end
     q="akash";
     w="akash";
     disp("q & w respectively are=");
     disp(q);
     disp(w);
     disp("comparing strings");
     x=strcmp(q,w);
```

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

Chapter 3

Stacks and Queues

Scilab code Exa 1.1.b example

```
//Exercise question 2:
//Implementing Push And Pop Functions:
function[y,sta1]=empty(sta)
  y=0;
  sta1=0;
  if(sta.top==0)
    y=0;
  else
    y=1;
  end
  sta1=sta
endfunction

function[sta]=push(stac,ele)
  sta=0;
  if(empty(stac)==0)
    stac.a=ele;
    stac.top=stac.top+1;
  else
    stac.a=[stac.a(:,:) ele]
    stac.top=stac.top+1;
  end
  disp(stac);
  sta=stac;
  funcprot(0)
endfunction

function[ele,sta]=pop(stack)
  ele='-1';
  if(empty(stack)==0)
    disp("Stack Underflow");
    break;
  else
    ele=stack.a(stack.top);
    stack.top=stack.top-1;
    if(stack.top~=0)
      b=stack.a(1);
    for i2=2:stack.top
      b=[b(:,:) stack.a(i2)];
    end
    stack.a=b;
  else
    stack.a='0';
  end
  end
  disp(stack);
  sta=stack;
endfunction
global stack
//Calling Routine:
stack=struct('a',0,'top',0);
stack=push(stack,4);
stack=push(stack,55);
stack=push(stack,199);
stack=push(stack,363);
[ele,stack]=pop(stack);
disp(stack,"After the above operations stack is:");
```

Scilab code Exa 3.1 example

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

Scilab code Exa 3.1.2 example

```
//Unolved Example 2:
clear all;
clc;
disp("Unsolved example 2");
//Implementing Stack using union:
function[stack]=sta_union(etype,a)
  stackelement=struct('etype',etype);
  [k,l]=size(a);
select stackelement.etype,
case 'int' then
  a=int32(a);
  stack=struct('top',l,'items',a);,
  case 'float' then
  a=double(a);
  stack=struct('top',1,'items',a);,
  case 'char' then
  a=string(a);
  stack=struct('top',1,'items',a);,
end
disp(stack,"Stack is:");
endfunction
a=[32 12.34 232 32.322]
stack=sta_union('float',a)
stack=sta_union('int',a)
stack=sta_union('char',a)
```

Scilab code Exa 1.2.b example

```
//Unsolved Example 1
clear all;
clc;
disp("example 3.7");
//To determine the syntacticaly valid string
function[l]=strlen(x)
  i=1;
  l=0;
  [j,k]=size(x)
  for i=1:k
    l=l+length(x(i));
  end
endfunction
function[]=stringvalid(str)
  str=string(str);
  stack=struct('a','0','top',0);
  l1=strlen(str);
  valid=1;
  l=1;
  while(l<=l1)
      if(str(l)=='('|str(l)=='['|str(l)=='{')
        if(stack.top==0)
          stack.a=str(l);
          stack.top=stack.top+1;
        else
          stack.a=[stack.a(:,:) str(l)];
          stack.top=stack.top+1;
        end
        disp(stack);
      end
      if(str(l)==')'|str(l)==']'|str(l)=='}')
        if(stack.top==0)
          valid=0;
          break;
        else
          i=stack.a(stack.top);
          b=stack.a(1);
          for i1=2:stack.top-1
            b=[b(:,:) stack.a(i1)]
          end
          stack.a=b;
         stack.top=stack.top-1;
         symb=str(l);
         disp(stack);
         if(((symb==')')&(i=='('))|((symb==']')&(i==
            '['))|((symb=='}')&(i=='{')))
        else
          valid=0;
          break;
        end
      end
    end
      l=l+1;
    end
    if(stack.top~=0)
      valid=0;
    end
    if(valid==0)
      disp("Invalid String");
    else
      disp("Valid String");
    end
  endfunction
  //Calling Routine:
  stringvalid(['(' 'A' '+' 'B' '}' ')'])
  stringvalid(['{' '[' 'A' '+' 'B' ']'  '-' '[' '('
     'C' '-'  'D' ')' ']'])
  stringvalid(['(' 'A' '+' 'B' ')' '-' '{' 'C' '+' '
     D' '}' '-' '[' 'F' '+' 'G' ']'])
  stringvalid(['(' '(' 'H' ')' '*' '{' '(' '[' 'J' '
     +' 'K' ']' ')' '}' ')'])
  stringvalid(['(' '(' '(' 'A' ')' ')' ')'])
```

Scilab code Exa 3.2 example

```
//example 3.2
//Queue  Operations
clear all;
clc;
function[q2]=push(ele,q1)
  if(q1.rear==q1.front)
    q1.a=ele;
    q1.rear=q1.rear+1;
  else
    q1.a=[q1.a(:,:) ele];
    q1.rear=q1.rear+1;
  end
  q2=q1;
endfunction
funcprot(0);
function[ele,q2]=pop(q1)
  ele=-1;
  q2=0;
  if(q1.rear==q1.front)
```

```
    disp("Queue Underflow");
    return;
  else
    ele=q1.a(q1.rear-q1.front);
    q1.front=q1.front+1;
    i=1;
    a=q1.a(1);
    for i=2:(q1.rear-q1.front)
      a=[a(:,:) q1.a(i)];
    end
    q1.a=a;
  end
  q2=q1;
endfunction
funcprot(0);
//Calling Routine:
q1=struct('a',0,'rear',0,'front',0)
q1=push(3,q1)
q1=push(22,q1);
q1=push(21,q1);
disp(q1,"Queue after insertion");
[ele,q1]=pop(q1)
disp(ele,"poped element");
disp(q1,"Queue after poping");
[ele,q1]=pop(q1);
[ele,q1]=pop(q1);
[ele,q1]=pop(q1);//Underflow Condition
```

Scilab code Exa 1.3.a example

```
clear all;
clc;
disp("Unsolved example 3");
function[l]=strlen(x)
  i=1;
  l=0;
  [j,k]=size(x)
  for  i=1:k
    l=l+length(x(i));
  end
endfunction
function []=str(st)
  stack=struct('a',0,'top',0);
  st=string(st);
  l=1;
  l1=strlen(st);
  symb=st(l);
  valid=1;
  while(l<l1)
    while(symb~='C')
      if(stack.top==0)
          stack.a=st(l);
          stack.top=stack.top+1;
       else
          stack.a=[stack.a(:,:) st(l)];
          stack.top=stack.top+1;
       end
       l=l+1;
       symb=st(l);
     end
     i=st(l+1);
    if(stack.top==0)
        valid=0;
        break;
     else
       symb1=stack.a(stack.top);
       stack.top=stack.top-1;
       if(i~=symb1)
         valid=0;
         break;
       end
     end
    l=l+1;
  end
    if(stack.top~=0)
      valid=0;
    end
    if(valid==0)
      disp("Not of the given format");
    else
      disp("String Of the Given Format");
    end
endfunction
//Calling Routine:
st=['A' 'A' 'B' 'A' 'C' 'A' 'B' 'A' 'A']
str(st)
st=['A' 'A' 'B' 'A' 'C' 'A' 'B' 'A' ]
str(st)
```

Scilab code Exa 3.3 example

```
//Solved Example 3.3:
//Convering an infix expression to a Postfix
   Expression:
function[sta]=push(stac,ele)
  sta=0;
  if(stac.top==0)
    stac.a=ele;
    stac.top=stac.top+1;
  else
    stac.a=[stac.a(:,:) ele]
    stac.top=stac.top+1;
  end
  disp(stac);
  sta=stac;
endfunction

function[ele,sta]=pop(stack)
```

```
  ele='-1';
  if(stack.top==0)
    disp("Stack Underflow");
    break;
  else
    ele=stack.a(stack.top);
    stack.top=stack.top-1;
    if(stack.top~=0)
      b=stack.a(1);
    for i2=2:stack.top
      b=[b(:,:) stack.a(i2)];
    end
    stack.a=b;
  else
    stack.a='0';
  end
  end
  sta=stack;
endfunction
function[l]=strlen(x)
  i=1;
  l=0;
  [j,k]=size(x)
  for  i=1:k
    l=l+length(x(i));
  end
endfunction
function[p]=pre(s1,s2)
     i1=0;
    select s1,
    case '+' then i1=5;
    case '-' then i1=5;
    case '*' then  i1=9;
    case '/' then i1=9;
    end
    i2=0;
    select s2,
    case '+' then i2=5;
    case '-' then i2=5;
    case '*' then  i2=9;
    case '/' then i2=9;
    end
    p=0;
    p=i1-i2;
    if(s1=='(')
      p=-1;
    end
    if(s2=='('&s1~=')')
      p=-1;
    end
    if(s1~='('&s2==')')
      p=1;
    end

  endfunction
function[a2]=intopo(a1,n)
  stack=struct('a',0,'top',0);
  l1=1;
  l2=strlen(a1(1))
  for i=2:n
    l2=l2+strlen(a1(i))
  end
  a2=list();
  while(l1<=l2)
    symb=a1(l1);
    if(isalphanum(string(a1(l1))))
      a2=list(a2,symb);
    else
      while(stack.top~=0&(pre(stack.a(stack.top),
         symb)>=0))
        [topsymb,stack]=pop(stack);
        if(topsymb==')'|topsymb=='(')
          a2=a2;
        else
          a2=list(a2,topsymb);
        end
        end
        if(stack.top==0|symb~=')')
          stack=push(stack,symb);
        else
          [ele,stack]=pop(stack);
        end
      end
      l1=l1+1;
    end
    while(stack.top~=0)
      [topsymb,stack]=pop(stack);
      if(topsymb==')'|topsymb=='(')
        a2=a2;
      else
        a2=list(a2,topsymb);
      end
    end
    disp(a2);
  endfunction
  //Calling Routine:
  a1=['(' '2' '+' '3' ')' '*' '(' '5' '-' '4' ')']
  a2=intopo(a1,11)
```

Chapter 4

Linked lists

Scilab code Exa 4.1 example

```
//List of words in a linked list.
clear all;
clc;
printf("\n Exapmle 4.1\n");
x=list('sci','lab','text','companionship','project')
   ;
disp("x=");
disp(x);
```

Scilab code Exa 4.2 example

```
//CIRCULAR LINKED LIST
clear all;
clc;
funcprot(0);
disp("Example 4.2");
function[link2]=append(ele,link1)
  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,,0,0)
      ;
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=1;
    link2(1)=link1(1)(1);
    else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      lin2=link1(1)(1);
      lin2.data=ele;
      lin2.add=link1(1)(1).add+1;
      link1(1)(1).nexadd=lin2.add;
      lin2.nexadd=link1(1)(1).add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      lin2=link1(1)(1);
      i=1;
      while(link1(i)(1).nexadd~=link1(1)(1).add)
        i=i+1;
      end
      j=i;
      lin2.data=ele;
      lin2.add=link1(i).add+1;
      lin2.nexadd=link1(1)(1).add;
      link1(i).nexadd=lin2.add;
      link2(1)=link1(1)(1);
      i=2;
      while(link1(i).nexadd~=lin2.add)
        link2(i)=(link1(i));
        i=i+1;
      end
      link2(i)=link1(i);
      link2(i+1)=lin2;
    end
  end
endfunction
```

```
function[link2]=add(ele,pos,link1);
   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,,0,0)
      ;
   i=1;
   while(i<=pos)
     if(link1(i).nexadd==link1(1)(1).add)
       break;
     else
       i=i+1;
     end
   end
   if(link1(i).nexadd~=link1(1)(1).add)
     i=i-1;
      lin2.data=ele;
      lin2.add=i;
      j=i;
      while(link1(j).nexadd~=link1(1)(1).add)
        link1(j).add=link1(j).add+1;
        link1(j).nexadd=link1(j).nexadd+1;
        j=j+1;
      end
      link1(j).add=link1(j).add+1;
      lin2.nexadd=link1(i).add;
      link1(i-1).nexadd=lin2.add;
      k=1;
      while(k<i)
         link2(k)=link1(k);
         k=k+1;
       end
       link2(k)=lin2;
       k=k+1;
       link2(k)=link1(k-1);
       k=k+1
       l=k-1;
       while(k~=j)
         link2(k)=link1(l);
         k=k+1;
         l=l+1;
       end
       link2(j)=link1(j-1);;
       link2(j+1)=link1(j);
     else
       if(i==pos)
         k=1;
         lin2.data=ele;
         lin2.add=link1(i-1).add+1;
         link1(i).add=link1(i).add+1;
         lin2.nexadd=link1(i).add;
         link1(i).nexadd=link1(1)(1).add;
         k=1;
         while(k<pos)
           link2(k)=link1(k);
           k=k+1;
         end
         link2(k)=lin2;
         link2(k+1)=link1(k)
       end
     end

endfunction
function[link2]=delete1(pos,link1)
  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,,0,0)
     ;
  i=1;
  j=1;
  while(i<pos)
    if((link1(j).nexadd==link1(1)(1).add))
      j=1;
      i=i+1;
    else
      i=i+1;
      j=j+1;
    end
  end
  if(link1(j).nexadd~=link1(1)(1).add)
    k=1;
    if(j==1)
      k=2;
      while(link1(k).nexadd~=link1(1)(1).add)
        link2(k-1)=link1(k);
        k=k+1;
      end
      link2(k-1)=link1(k);
      link2(k-1).nexadd=link2(1).add;
    else
      lin2=link1(j);
      link1(j-1).nexadd=link1(j+1).add;
      k=1;
      while(link1(k).nexadd~=link1(j+1).add)
        link2(k)=link1(k);
        k=k+1;
      end
      link2(k)=link1(k);
      k=k+2;
      while(link1(k).nexadd~=link1(1)(1).add)
        link2(k-1)=link1(k);
        k=k+1;
      end
      link2(k-1)=link1(k);
    end
  else
    link1(j-1).nexadd=link1(1)(1).add;
    l=1;
    while(link1(l).nexadd~=link1(1)(1).add)
      link2(l)=link1(l);
      l=l+1;
    end
    link2(l)=link1(l);
  end
endfunction
//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);
```

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

Scilab code Exa 4.3 example

```
//List Insertion.
clc;
clear all;
disp("Example 4.3");
funcprot(0)
function[link2]=insert_pri(ele,link1)
  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,,0,0)
     ;
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=1;
    link2(1)=link1(1)(1);
  else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      if(ele>=link1(1)(1).data)
        t=ele;
        p=link1(1)(1).data;
      else
        t=link1(1)(1).data;
        p=ele;
      end
      link1(1)(1).data=t;
      lin2=link1(1)(1);
      lin2.data=p;
      lin2.add=2;
      lin2.nexadd=link1(1)(1).add;
      link1(1)(1).nexadd=lin2.add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      i=1;
      a=[];
      while(link1(i).nexadd~=link1(1)(1).add)
        a=[a(:,:) link1(i).data];
        i=i+1;
      end
      a=[a(:,:) link1(i).data];
      a=gsort(a);
      j=1;
      while(j<=i)
        link1(j).data=a(j);
        j=j+1;
      end
      k=1;
      while(link1(k).data>=ele)
        if(link1(k).nexadd==link1(1)(1).add)
          break;
        else
          link2(k)=link1(k);
          k=k+1;
        end
      end
       if(link1(k).nexadd~=link1(1)(1).add)
         lin2=link1(k);
         lin2.data=ele;
         lin2.add=link1(k).add;
         j=k;
         y=link1(1)(1).add;
         while(link1(k).nexadd~=y)
           link1(k).add=link1(k).add+1;
```

```
          link1(k).nexadd=link1(k).nexadd+1;
          k=k+1;
        end
        link1(k).add=link1(k).add+1;
        lin2.nexadd=link1(j).add;
        link2(j)=lin2;
        j=j+1;
        while(j<=k+1)
          link2(j)=link1(j-1);
          j=j+1;
        end
      else
        lin2=link1(k);
        lin2.data=ele;
        lin2.nexadd=link1(1)(1).add;
        lin2.add=link1(k).add+1;
        link1(k).nexadd=lin2.add;
        j=1;
        while(j<=k)
          link2(j)=link1(j);
          j=j+1;
        end
        link2(j)=lin2;
      end
    end
  end
endfunction
//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);
link1=insert_pri(3,link1);
link1=insert_pri(4,link1);
link1=insert_pri(22,link1);
link1=insert_pri(21,link1);
link1=insert_pri(11,link1);
disp(link1,"List After Insertions");
```

Scilab code Exa 4.4 example

```
//Deletion from the list:
function[link2]=append(ele,link1)
  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,,0,0)
     ;
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=0;
    link1(1)(1).prevadd=0;
    link2(1)=link1(1)(1);
    else
    if(link1(1)(1).nexadd==0)
      lin2=link1(1)(1);
      lin2.data=ele;
      lin2.add=link1(1)(1).add+1;
      link1(1)(1).nexadd=lin2.add;
      lin2.nexadd=0;
      lin2.prevadd=link1(1)(1).add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      lin2=link1(1)(1);
      i=1;
      while(link1(i)(1).nexadd~=0)
        i=i+1;
      end
      j=i;
      lin2.data=ele;
      lin2.add=link1(i).add+1;
      lin2.nexadd=0;
      link1(i).nexadd=lin2.add;
      lin2.prevadd=link1(i).add;
      link2(1)=link1(1)(1);
      i=2;
      while(link1(i).nexadd~=lin2.add)
        link2(i)=(link1(i));
        i=i+1;
      end
      link2(i)=link1(i);
      link2(i+1)=lin2;
    end
  end
endfunction
function[link2]=add(ele,pos,link1);
   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,,0,0)
      ;
   i=1;
   while(i<=pos)
     if(link1(i).nexadd==0)
       break;
     else
       i=i+1;
     end
   end
   if(link1(i).nexadd~=0)
     i=i-1;
      lin2.data=ele;
      lin2.add=i;
      j=i;
      while(link1(j).nexadd~=0)
        link1(j).prevadd=link1(j).prevadd+1;
        link1(j).add=link1(j).add+1;
        link1(j).nexadd=link1(j).nexadd+1;
        j=j+1;
      end
      link1(j).prevadd=link1(j).prevadd+1;
      link1(j).add=link1(j).add+1;
      lin2.nexadd=link1(i).add;
```

```
        link1(i).prevadd=lin2.add;
        lin2.prevadd=link1(i-1).add;
        link1(i-1).nexadd=lin2.add;
        k=1;
        while(k<i)
           link2(k)=link1(k);
           k=k+1;
         end
         link2(k)=lin2;
         k=k+1;
         link2(k)=link1(k-1);
         k=k+1
         l=k-1;
         while(k~=j)
           link2(k)=link1(l);
           k=k+1;
           l=l+1;
         end
         link2(j)=link1(j-1);;
         link2(j+1)=link1(j);
       else
         if(i==pos)
           k=1;
           lin2.data=ele;
           lin2.add=link1(i-1).add+1;
           link1(i).add=link1(i).add+1;
           lin2.nexadd=link1(i).add;
           link1(i).prevadd=lin2.add;
           lin2.prevadd=link1(i-1).add;
           k=1;
           while(k<pos)
             link2(k)=link1(k);
             k=k+1;
           end
           link2(k)=lin2;
           link2(k+1)=link1(k)
         end
       end
```

```

endfunction
function[link2]=delete1(pos,link1)
  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,,0,0)
     ;
  i=1;
  while(i<=pos)
    if((link1(i).nexadd==0))
      break;
    else
      i=i+1;
    end
  end
  if(link1(i).nexadd~=0)
    i=i-1;
    j=1;
    if(i==1)
      j=1;
      while(link1(j).nexadd~=0)
        link2(j)=link1(j);
        j=j+1;
      end
      link2(j)=link1(j);
    else
      link1(i-1).nexadd=link1(i+1).add;
      link1(i+1).prevadd=link1(i-1).add;
    while(link1(j).nexadd~=link1(i+1).add)
      link2(j)=link1(j);
      j=j+1;
    end
    if(j~=i-1)
      link2(j)=link1(j);
      link2(j+1)=link1(j+1);
      k=i+1;
      l=2;
    else
      link2(j)=link1(j);
```

```
        k=i+1;
        l=1;
      end
      while(link1(k).nexadd~=0)
        link2(j+l)=link1(k);
        k=k+1;
        l=l+1;
      end
      link2(j+l)=link1(k);
    end
    else
      if(i==pos)
        j=1;
        link1(i-1).nexadd=0;
        while(j<=i-1)
          link2(j)=link1(j);
          j=j+1;
        end
      end
    end
endfunction
//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);
link1=append(4,link1);
link1=append(6,link1);
link1=add(10,2,link1);
link1=delete1(3,link1);
disp(link1,"After the above manipulation the list is
   ");
```

Chapter 5

Trees

Scilab code Exa 5.1 example

```

funcprot(0);
function[tree]=maketree(x)
  tree=zeros(30,1);
  for i=1:30
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[tree1]=setleft(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j)
    tree(2*j)=x;
  else
    tree(2*j)=x;
    tree(2*j+1)=-2;
    for l=i:2*j-1
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[tree1]=setright(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j+1)
    tree(2*j+1)=x;
  else
    tree(2*j+1)=x;
    tree(2*j+2)=-2;
    for l=i:2*j
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[x]=isleft(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j)
    if((tree(2*j)~=-1)|(tree(2*j)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
function[x]=isright(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j+1)
    if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
//Calling Routine:
tree=maketree(3);
disp(tree,"Tree made");
tree=setleft(tree,3,1);
disp(tree,"After setting 1 to left of 3");
```

```
tree=setright(tree,3,2);
disp(tree,"After setting 2 to right of 3");
tree=setright(tree,2,4);
tree=setleft(tree,2,5);
tree=setright(tree,1,6);
tree=setright(tree,5,8);
disp(tree,"After above operations:");
x=isright(tree,3);
disp(x,"Checking for the right son of 3 yes if 1
   else no");
x=isleft(tree,2);
disp(x,"Check for left son of 2");
```

Scilab code Exa 5.2 example

```
funcprot(0);
function[tree]=maketree(x)
  tree=zeros(30,1);
  for i=1:30
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[tree1]=setleft(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j)
    tree(2*j)=x;
  else
    tree(2*j)=x;
    tree(2*j+1)=-2;
    for l=i:2*j-1
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[tree1]=setright(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j+1)
    tree(2*j+1)=x;
  else
    tree(2*j+1)=x;
    tree(2*j+2)=-2;
    for l=i:2*j
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[x]=isleft(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j)
    if((tree(2*j)~=-1)|(tree(2*j)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
function[x]=isright(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j+1)
    if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
funcprot(0);
function[]=inorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
```

```
    inorder(tree,2*p);
    printf("%d\t",tree(p));
    inorder(tree,2*p+1);
  end
endfunction
function[]=preorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    printf("%d\t",tree(p));
    preorder(tree,2*p);
    preorder(tree,2*p+1);
  end
endfunction
function[]=postorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    postorder(tree,2*p);
    postorder(tree,2*p+1);
    printf("%d\t",tree(p));
  end
endfunction
//Calling Routine:
tree=maketree(3);
tree=setleft(tree,3,1);
tree=setright(tree,3,2);
tree=setleft(tree,2,4);
tree=setright(tree,2,5);
disp("Inorder traversal");
inorder(tree,1);
disp("Preorder traversal");
preorder(tree,1);
disp("Postorder traversal");
postorder(tree,1);
```

Scilab code Exa 5.3 example

```
funcprot(0);
function[tree]=maketree(x)
  tree=zeros(1,30);
  for i=1:30
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[tree1]=setleft(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j)
    tree(2*j)=x;
  else
    tree(2*j)=x;
    tree(2*j+1)=-2;
    for l=i:2*j-1
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[tree1]=setright(tree,tre,x)
  tree1=[];
  i=1;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>2*j+1)
    tree(2*j+1)=x;
  else
    tree(2*j+1)=x;
    tree(2*j+2)=-2;
    for l=i:2*j
      tree(i)=-1;
    end
  end
  tree1=tree;
endfunction
function[x]=isleft(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j)
    if((tree(2*j)~=-1)|(tree(2*j)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
```

```
function[x]=isright(tree,tre)
  i=1;
  x=0;
  while(tree(i)~=-2)
    if(tree(i)==tre)
      j=i;
    end
    i=i+1;
  end
  if(i>=2*j+1)
    if((tree(2*j+1)~=-1)|(tree(2*j+1)~=-2))
      x=1;
      return 1;
    else
      return 0;
    end
  else
    x=0;
    return x;
  end
endfunction
funcprot(0);
function[]=inorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    inorder(tree,2*p);
    disp(tree(p),"  ");
    inorder(tree,2*p+1);
  end
endfunction
function[]=preorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    disp(tree(p),"  ");
    preorder(tree,2*p);
    preorder(tree,2*p+1);
  end
endfunction
function []=postorder(tree,p)
  if(tree(p)==-1|tree(p)==-2)
    return;
  else
    postorder(tree,2*p);
    postorder(tree,2*p+1);
    disp(tree(p),"  ");
  end
endfunction
function [tree1]=binary(tree,x)
  p=1;
  while(tree(p)~=-1&tree(p)~=-2)
    q=p;
    if(tree(p)>x)
      p=2*p;
    else
      p=2*p+1;
    end
  end
  i=1;
  while(tree(i)~=-2)
    i=i+1;
  end
  if(tree(q)>x)
    if(i==2*q)
      tree(2*q)=x;
      tree(2*q+1)=-2
    else
      if(i<2*q)
        tree(i)=-1;
        tree(2*q+1)=-2;
        tree(2*q)=x;
      end
    end

  else
```

```
    if(i==2*q+1)
      tree(2*q+1)=x;
      tree(2*q+2)=-2;
    else
      if(i<2*q+1)
        tree(i)=-1;
        tree(2*q+1)=x;
        tree(2*q+2)=-2;
      end
    end

  end
  tree1=tree;
endfunction
//Calling Routine:
tree=maketree(3);
tree=binary(tree,1);
tree=binary(tree,2);
tree=binary(tree,4);
tree=binary(tree,5);
disp(tree,"Binary tree thus obtaine by inserting
   1,2,4and5 in tree rooted 3 is:");
```

Scilab code Exa 5.4 example

```
function[tree1]=binary(tree,x)
  p=1;
  while(tree(p)~=-1&tree(p)~=-2)
    q=p;
    if(tree(p)>x)
      p=2*p;
    else
      p=2*p+1;
    end
  end
  if(tree(q)>x)
    if(tree(2*q)==-2)
      tree(2*q)=x;
      tree(2*q+1)=-2;
    else
      tree(2*q)=x;
    end
  else
    if(tree(2*q+1)==-2)
      tree(2*q+1)=x;
      tree(2*q+2)=-2;
    else
      tree(2*q+1)=x;
    end
  end
  tree1=tree;
endfunction
funcprot(0);
function[tree]=maketree(x)
  tree=zeros(40,1);
  for i=1:40
    tree(i)=-1;
  end
  tree(1)=x;
  tree(2)=-2;
endfunction
function[]=duplicate1(a,n)
  tree=maketree(a(1));
  q=1;
  p=1;
  i=2;
  x=a(i)
  while(i<n)
    while(tree(p)~=x&tree(q)~=-1&tree(q)~=-2)
      p=q;
      if(tree(p)<x)
        q=2*p;
      else
```

```
        q=2*p+1;
      end
    end
    if(tree(p)==x)
      disp(x," Duplicate  ");
    else
      tree=binary(tree,x);
    end
    i=i+1;
    x=a(i);
  end
  while(tree(p)~=x&tree(q)~=-1&tree(q)~=-2)
      p=q;
      if(tree(p)<x)
        q=2*p;
      else
        q=2*p+1;
      end
    end
    if(tree(p)==x)
      disp(x," Duplicate  ");
    else
      tree=binary(tree,x);
    end
endfunction
//Calling Adress:
a=[22 11 33 22 211 334]
duplicate1(a,6)
a=[21 11 33 22 22 334]
duplicate1(a,6)
```

Chapter 6

Graphs

Scilab code Exa 6.1 example

```
clear all;
clc;
disp("Example 6.1");
//Depth First Search Traversal
funcprot(0)
function[]=Dfs(adj,n);
  i=1,j=1;
  colour=[];
  for i=1:n
  for j=1:n
      colour=[colour(:,:) 0];
    end
  end
  disp("The DFS traversal is");
dfs(adj,colour,1,n);
endfunction
function[]=dfs(adj,colour,r,n)
  colour(r)=1;
  disp(r,"  ");
  for i=1:n
    if(adj((r-1)*n+i)&(colour(i)==0))
```

```
      dfs(adj,colour,i,n);
    end
  end
  colour(r)=2;
endfunction
//Calling Routine:
n=4;
adj=[0 1 1 0 0 0 0 1 0 0 0 1 0 0 0 0]
Dfs(adj,n)
```

Scilab code Exa 6.2 example

```
clear all;
clc;
disp("Example 6.2");
////BFS Traversal
funcprot(0)
function[q2]=push(ele,q1)
  if(q1.rear==q1.front)
    q1.a=ele;
    q1.rear=q1.rear+1;
  else
    q1.a=[q1.a(:,:) ele];
    q1.rear=q1.rear+1;
  end
  q2=q1;
endfunction
function[ele,q2]=pop(q1)
  ele=-1;
  q2=0;
  if(q1.rear==q1.front)
        return;
  else
    ele=q1.a(q1.rear-q1.front);
    q1.front=q1.front+1;
    i=1;
    a=q1.a(1);
    for i=2:(q1.rear-q1.front)
      a=[a(:,:) q1.a(i)];
    end
    q1.a=a;
  end
  q2=q1;
endfunction

function[]=Bfs(adj,n);
  i=1,j=1;
  colour=[];
  for i=1:n
  for j=1:n
      colour=[colour(:,:) 0];
    end
  end
  disp("The BFS Traversal is");
bfs(adj,colour,1,n);
endfunction
function[]=bfs(adj,colour,s,n)
  colour(s)=1;
  q=struct('rear',0,'front',0,'a',0);
  q=push(s,q);
  while((q.rear)-(q.front)>0)
    [u,q]=pop(q);
    disp(u," ");
    for i=1:n
      if(adj((u-1)*n+i)&(colour(i)==0))
        colour(i)=1;
        q=push(i,q);
      end
    end
    colour(u)=2;
  end
endfunction
//Calling Routine:
```

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

Scilab code Exa 6.3 example

```
clear all;
clc;
disp("Example 6.3");
//Warshall's Algorithm
clc;
clear all;
funcprot(0)
function[path]=transclose(adj,n)
  for  i=1:n
    for j=1:n
      path((i-1)*n+j)=adj((i-1)*n+j);
    end
  end
  for k=1:n
    for  i=1:n
      if(path((i-1)*n+k)==1)
        for j=1:n
          path((i-1)*n+j)=path((i-1)*n+j)|path((k-1)
              *n+j);
        end
      end
    end
  end
  printf("Transitive closure for the given graph is
      :\n");
  for  i=1:n
    printf("For vertex %d \n",i);
    for j=1:n
      printf("%d %d is %d\n",i,j,path((i-1)*n+j));
    end
  end
endfunction
//Calling Routine:
n=3;
adj=[0 1 0 0 0 1 0 0 0]
path=transclose(adj,n)
```

Scilab code Exa 6.4 example

```
clear all;
clc;
disp("Example 6.4");
//Finnding Transitive Closure
funcprot(0);
function[path]=Tranclose(adj,n);
  i=1,j=1;
  path=zeros(n*n,1);
  path=tranclose(adj,n);
  printf("Transitive Closure Of Given Graph is:\n");
  for i=1:n
    printf("For Vertex %d\n",i);
    for j=1:n
      printf(" %d %d is %d\n",i,j,path((i-1)*n+j));
    end
  end

endfunction
function[path]=tranclose(adj,n)
  adjprod=zeros(n*n,1);
  k=1;
  newprod=zeros(n*n,1);
  for i=1:n
    for j=1:n
      path((i-1)*n+j)=adj((i-1)*n+j);
      adjprod((i-1)*n+j)= path((i-1)*n+j);
    end
  end
  for i=1:n
    newprod=prod1(adjprod,adj,n);
    for j=1:n
      for k=1:n
        path((j-1)*n+k)=path((j-1)*n+k)|newprod((j
           -1)*n+k);
      end
    end
    for j=1:n
      for k=1:n
        adjprod((j-1)*n+k)=newprod((j-1)*n+k);
      end
    end
  end
endfunction
function[c]=prod1(a,b,n)
  for i=1:n
    for j=1:n
      val=0
      for k=1:n
        val=val|(a((i-1)*n+k)&b((k-1)*n+j));
      end
      c((i-1)*n+j)=val;
    end
  end
endfunction
//Calling Routine:
n=3;
adj=[0 1 0 0 0 1 0 0 0]
path=Tranclose(adj,n)
```

Scilab code Exa 6.5 example

```
clear all;
clc;
disp("Example 6.5");
//Finding The  Number Of Simple Paths From One Point
   To Another In A Given Graph
funcprot(0)
function []=sim_path(n,adj,i,j);
  l=0;
  m=1;
  for m=1:n
    l=l+path(m,n,adj,i,j);
  end
  printf("There are %d  simple paths from %d to %d
     in the given graph\n",l,i,j);
endfunction
function[b]=path(k,n,adj,i,j)
  b=0;
  if(k==1)
    b=adj((i-1)*n+j);
  else
    for c=1:n
      if(adj((i-1)*n+c)==1)
        b=b+path(k-1,n,adj,c,j);
      end
    end
  end
  return b;
endfunction
n=3;
adj=[0 1 1 0 0 1 0 0 0];
b=sim_path(n,adj,1,3)
```

Scilab code Exa 6.6 example

```
clear all;
```

```
clc;
disp("Example 6.6");
//Dijkstras Algorithm
funcprot(0)
function[l]=short(adj,w,i1,j1,n)
  for i=1:n
    for j=1:n
      if(w((i-1)*n+j)==0)
        w((i-1)*n+j)=9999;
      end
    end
  end

  distance=[];
  perm=[];
  for i=1:n
    distance=[distance(:,:) 99999];
    perm=[perm(:,:) 0];
  end
  perm(i1)=1;
  distance(i1)=0;
  current=i1;
  while(current~=j1)
    smalldist=9999;
    dc=distance(current);
    for i=1:n
      if(perm(i)==0)
        newdist=dc+w((current-1)*n+i);
        if(newdist<distance(i))
          distance(i)=newdist;
        end
        if(distance(i)<smalldist)
          smalldist=distance(i);
          k=i;
        end
      end
    end
    current=k;
    perm(current)=1;
  end
  l=distance(j1);
  printf("The  shortest path between %d and %d is %d
     ",i1,j1,l);
endfunction
//Calling Routine:
n=3;
adj=[0 1 1 0 0 1 0 0 0]//Adjacency  List
w=[0 12 22 0 0 9 0 0 0]//weight list fill 0 for no
   edge
short(adj,w,1,3,n);
```

Scilab code Exa 6.7 example

```
clear all;
clc;
disp("Example 6.7");
//Finding The Number Of Paths From One Vertex To
   Another Of A Given Length

function[b]=path(k,n,adj,i,j)
  b=0;
  if(k==1)
    b=adj((i-1)*n+j);
  else
    for c=1:n
      if(adj((i-1)*n+c)==1)
        b=b+path(k-1,n,adj,c,j);
      end
    end
  end
    printf("Number of paths from vertex %d to %d of
       length %d are %d",i,j,k,b);
  return b;
```

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

Chapter 7

Sorting

Scilab code Exa 7.1 example

```
clear all;
clc;
disp("Example 7.1");
funcprot(0);
function[a1]=insertion(a,n)
  for k=1:n
    y=a(k);
    i=k;
    while(i>=1)
      if(y<a(i))
        a(i+1)=a(i);
        a(i)=y;
      end
      i=i-1;
    end
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[5 4 3 2 1]        // worst-case behaviour of
```

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

Scilab code Exa 7.2 example

```
clear all;
clc;
disp("Example 7.2");
funcprot(0);
function[a1]=insertion(a,n)
  for k=1:n
    y=a(k);
    i=k;
    while(i>=1)
      if(y<a(i))
        a(i+1)=a(i);
        a(i)=y;
      end
      i=i-1;
    end
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[2 3 4 5 1]
disp(a,"Given Array");
a1=insertion(a,5)
```

Scilab code Exa 7.3 example

```
clear all;
```

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

Scilab code Exa 7.4 example

```
clear all;
clc;
disp("Example 7.4");
function[a1]=insertion(a,n)
  for k=1:n
    y=a(k);
    i=k;
    while(i>=1)
      if(y<a(i))
        a(i+1)=a(i);
        a(i)=y;
      end
      i=i-1;
    end
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[3 1 2]
disp(a,"Given Array");
a1=insertion(a,3)
```

Scilab code Exa 7.5 example

```
clear all;
clc;
disp("Example 7.5");
funcprot(0);
function[a1]=mergesort(a,p,r)
  if(p<r)
    q=int((p+r)/2);
    a=mergesort(a,p,q);
    a=mergesort(a,q+1,r);
    a=merge(a,p,q,r);
  else
    a1=a;
    return;
  end
  a1=a;
endfunction
function[a1]=merge(a,p,q,r)
  n1=q-p+1;
  n2=r-q;
  left=zeros(n1+1);
  right=zeros(n2+1);
  for i=1:n1
    left(i)=a(p+i-1);
  end
  for i1=1:n2
      right(i1)=a(q+i1);
  end
  left(n1+1)=999999999;
  right(n2+1)=999999999;
  i=1;
  j=1;
  k=p;
  for  k=p:r
    if(left(i)<=right(j))
      a(k)=left(i);
      i=i+1;
    else
      a(k)=right(j);
      j=j+1;
    end
  end
  a1=a;
endfunction
//Calling Routine:
a=[26 5 77 1 61 11 59 15 48 19]
disp(a,"Given Array");
a1=mergesort(a,1,10)
disp(a1,"Sorted array is:");
```

Scilab code Exa 7.6 example

```
clear all;
clc;
disp("Example 7.7");
function[a1]=shell(a,n,incr,nic)
  for i=1:nic
    span=incr(i);
    for j=span+1:n
      y=a(j);
      k=j-span;
      while(k>=1&y<a(k))
```

```
          a(k+span)=a(k);
        k=k-span;
      end
      a(k+span)=y;
    end
  end
  a1=a;
  disp(a1,"Sorted array is:");
endfunction
//Calling Routine:
a=[23 21 232 121 2324 1222433 1212]
disp(a,"Given Array");
incr=[5 3 1]//must always end with 1
a1=shell(a,7,incr,3)
```

Scilab code Exa 7.7 example

```
clear all;
clc;
disp("Example 7.7");
function[a1]=shell(a,n,incr,nic)
  for i=1:nic
    span=incr(i);
    for j=span+1:n
      y=a(j);
      k=j-span;
      while(k>=1&y<a(k))
          a(k+span)=a(k);
        k=k-span;
      end
      a(k+span)=y;
    end
  end
  a1=a;
  disp(a1,"Sorted array is:");
```

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

Scilab code Exa 7.8 example

```
clear all;
clc;
function[]=sortedsearch(a,n,ele)
  if(a(1)>ele|a(n)<ele)
    disp("NOT IN THE LIST");
  else
    i=1;
    j=0;
    for i=1:n
      if(a(i)==ele)
        printf("FOUND %d AT %d",ele,i);
        j=1;
      else
        if(a(i)>ele)
          break;
        end
      end
    end
    if(j==0)
      disp("%d NOT FOUND",ele);
    end
  end
endfunction
//Calling Routine:
a=[2 22 23 33 121 222 233]//a should be  sorted
disp(a,"Given array");
```

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

Chapter 8

Hashing

Scilab code Exa 8.1 example

```
clear all;
clc;
disp("Example 8.1");
k=12320324111220;
p1=123;
p2=203;
p3=241;......//key k partitioned into parts that are
    3 decimal long.
p4=112;
p5=20;
///.....using shift folding...
//....partitions are added to get the hash address.
z=p1+p2+p3+p4+p5;
disp(z);
//when folding at the boundaries is used,we reverse
   p2 and p4.
 p2=302;
 p4=211;
 x=p1+p2+p3+p4+p5;
 disp(x);
```

Scilab code Exa 8.2 example

```
clear all;
clc;
disp("Example 8.2");
function []=stringtoint()
    num= ascii("scilab");
    disp("displayin ascii codes of alphabets=");
    disp(num);
    // converting strings into unique non-negative
       integer and suming these unique integers.
    z=sum(num);
    disp("displayin sum of these integers");
    disp(z);
endfunction
stringtoint()
```

Chapter 9

Priority Queues

Scilab code Exa 9.1 example

```
clear all;
clc;
//Implementing Priority Queue Using Lists
funcprot(0)
function[link2]=insert_pri(ele,link1)
  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,,0,0);
  if(link1(1)(1).add==0)
    link1(1)(1).data=ele;
    link1(1)(1).add=1;
    link1(1)(1).nexadd=1;
    link2(1)=link1(1)(1);
  else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      if(ele>=link1(1)(1).data)
        t=ele;
        p=link1(1)(1).data;
      else
        t=link1(1)(1).data;
        p=ele;
      end
      link1(1)(1).data=t;
      lin2=link1(1)(1);
      lin2.data=p;
      lin2.add=2;
      lin2.nexadd=link1(1)(1).add;
      link1(1)(1).nexadd=lin2.add;
      link2(1)=link1(1)(1);
      link2(2)=lin2;
    else
      i=1;
      a=[];
      while(link1(i).nexadd~=link1(1)(1).add)
        a=[a(:,:) link1(i).data];
        i=i+1;
      end
      a=[a(:,:) link1(i).data];
      a=gsort(a);
      j=1;
      while(j<=i)
        link1(j).data=a(j);
        j=j+1;
      end
      k=1;
      while(link1(k).data>=ele)
        if(link1(k).nexadd==link1(1)(1).add)
          break;
        else
          link2(k)=link1(k);
          k=k+1;
        end
      end
       if(link1(k).nexadd~=link1(1)(1).add)
         lin2=link1(k);
         lin2.data=ele;
         lin2.add=link1(k).add;
         j=k;
         y=link1(1)(1).add;
```

```
          while(link1(k).nexadd~=y)
            link1(k).add=link1(k).add+1;
            link1(k).nexadd=link1(k).nexadd+1;
            k=k+1;
          end
          link1(k).add=link1(k).add+1;
          lin2.nexadd=link1(j).add;
          link2(j)=lin2;
          j=j+1;
          while(j<=k+1)
            link2(j)=link1(j-1);
            j=j+1;
          end
        else
          lin2=link1(k);
          lin2.data=ele;
          lin2.nexadd=link1(1)(1).add;
          lin2.add=link1(k).add+1;
          link1(k).nexadd=lin2.add;
          j=1;
          while(j<=k)
            link2(j)=link1(j);
            j=j+1;
          end
          link2(j)=lin2;
        end
      end
    end
  endfunction
  function[ele,link2]=extract_min(link1);
    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,,0,0)
       ;
    i=1;
    ele=-1;
    if(link1(1)(1).add==0)
      disp("Underflow");
      return;
  else
    if(link1(1)(1).nexadd==link1(1)(1).add)
      link1(1)(1).add=0;
      link1(1)(1).nexadd=0;
      ele=link1(1)(1).data;
      link1(1)(1).data=0;
      link2(1)=link1(1)(1);
    else
      i=1;
      while(link1(i).nexadd~=link1(1)(1).add)
        link2(i)=link1(i);
        i=i+1;
      end
      ele=link1(i).data;
      link2(i-1).nexadd=link2(1).add;
    end
  end
endfunction
//Calling Routine:
link1=struct('data',0,'add',0,'nexadd',0);
link1=insert_pri(3,link1);
link1=insert_pri(4,link1);
link1=insert_pri(22,link1);
link1=insert_pri(21,link1);
link1=insert_pri(11,link1);
disp(link1,"List After Insertions");
[ele,link1]=extract_min(link1)
disp(ele,"Element after the min extraction");
```