1 Sorting of Strings

1:start 2:Declare str[20][25] and temp[25]
3:Read n 4:i=0
5:Repeat steps 6 to 7 while i<=n
6:Read str[i] 7:i=i+1
8:i=0
9:Repat steps 10 to while i<=n
10:j=i+1
11:Repeat steps 12 to 13 while j<=n
12:If str[i]>str[j]
temp=str[i]
str[i]=str[j]
str[j]=temp
13:j=j+1 14:i=i+1
15:Print "Sorted array is:"
16:i=0
17:Repeat steps 18 to 19 while i<=n
18:Print str[i] 19:i++ 20:stop

1 Sorting of string

```
#include <stdio.h>
#include <string.h> #include <conio.h>
void main()
{    int i, j,n;
char str [20][25],temp[25];
printf("\n How many strings? \n");
Scanf("%d", &n);
printf("\n Enter the strings \n");
for (i=0; i<= n; i++)
        gets(str[i]);
for (i=0;i<=n;i++)
        for (j=i+1; j<= n; j++)
if (strcmp (str [i], str[j])>0)
{  strcpy (temp, str[i]);
       strcpy (str[i], str[j]);
       strcpy (str[j], temp);  }
printf("\n Sorted Array:");
for(i=0;i<=n;i++)
puts(str[i]);
getch( );   }
```

2 String reversing pointers

main() :-

1:start 2:declare str[15]
3:read a string into str
4:call stringrev(str)
5:print the reverse 6:stop

*String rev(*s):-*

1:start
2:set p=s and l=0
3:Repeat l=l+1 and p=p+1 while *p≠NULL
4:p=s and q=s+l-1
5:Repeat step 6 and 7 while p<q
6: +=*p
*p=*q
*q=t
7: p=p+1 and q=q-1
8: Return

2 String reverse

```
#include<stdio.h>
#include<conio.h>

void stringrev(char*);
void main()
{   char str[15];
clrscr();
printf("Enter a string : ");
gets(str);
stringrev(str);
printf("\nReverse is %s",str);
getch();   }

void stringrev(char *s)
{   int l;
char *p,*q,t;
for(p=s,l=0;*p!='\0';l++,p++)
;
for(p=s,q=s+l-1;p<q;p++,q--)
{   t=*p;
*p=*q;
*q=t;   }  }
```

3 Pattern Matching

1:start
2:Read a string into t
3:Read a pattern into p
4:set l1=length of t
5:set l2=length of p 6:i=0
7:Repeat steps 8 to 13 while i<=l1-l2
8:j=0 and k=i
9:Repeat steps 10 to 11 while j<l2
10:If(t[k]≠p[j]) goto step 12
11:j=j+1 & k=k+1
12:if(j==l2) goto step 14
13:i=i+1
14:if(i>l1-l2)
print pattern not matching...
else
print string pattern present at index i+1
15:stop

3 Pattern Matching

```
#include<stdio.h>
#include<conio.h>
#include<string.h>

void main()
{   char t[25],p[25];
int l1,l2,i,j,k;
clrscr();
printf("Enter a text : ");
gets(t);
printf("\nEnter the pattern : ");
gets(p);
l1=strlen(t);
l2=strlen(p);
for(i=0;i<=l1-l2;i++)
{  for(j=0,k=i;j<l2;j++,k++)
if(t[k]!=p[j]) break;
if(j==l2) break;  }
if(i>l1-l2) printf("\nPattern not matching...");
else printf("\nPattern found matching at location %d",i+1);
getch();   }
```

4 Search 2D array

1:start 2:found=0
3:Read dimensions m&n
4:i=0
5:Repeat steps 6 to 10 while i<m
6:j=0
7:Repeat steps 8 to 9 while j<n
8:Read a[i][j] 9:j=j+1
10:i=i+1 11:i=0
12:Repeat steps 13 to 17 while i<m
13:j=0
14:Repeat steps 15 to 16 while j<n
15:print a[i][j]
16:j=j+1 17:i=i+1
18:Read x 19:i=0
20:Repeat steps 21 to 25 while i<m
21:j=0
22:Repeat steps 23 to 24 while j<n
23:If(a[i][j]==x) , found=1 then goto step 26
24:j=j+1 25:i=i+1
26:If(found)
print x is found at row no i+1 and column number j+1
else
print x is not found
27:stop

4 Search 2D Array

```
#include<stdio.h>
#include<conio.h>

void main()
{  int a[10][10],i,j,m,n,x,found=0;
clrscr();
printf("\nEnter the dimension : ");
scanf("%d%d",&m,&n);
printf("\nEnter %d elements row wise : ",m*n);
for(i=0;i<m;i++)      for(j=0;j<n;j++)
scanf("%d",&a[i][j]);
printf("\nElements in the array:\n");
for(i=0;i<m;i++)
{   for(j=0;j<n;j++)
printf("%4d",a[i][j]);
printf("\n");   }
printf("\nEnter element to search : ");
scanf("%d",&x);
for(i=0;i<m;i++)        for(j=0;j<n;j++)
if(a[i][j]==x)
{   found=1;
goto label;   }
label:if(found)
printf("\n%d is found at row no.%d, column no.%d",x,i+1,j+1);
else printf("\n%d is not found",x);
getch();    }
```

5 Append 2 Arrays

1:start 2:read m
3:initialize i=0
4:repeat steps 5 to 6 while i<m
5:read a[i] 6:i=i+1
7:read n 8:initialize i=0
9:repeat steps 10 to 11 while i<n
10:read b[i] 11:i=i+1
12:initialize i=m,j=0
13:repeat steps 14 to 15 while j<n
14:set a[i]=b[i] 15:1=i+1,j=j+1
16:initialize i=0
17:repeat steps 18 to 19 while i<m+n
18:print a[i] 19:i=i+1 20:stop

5 Append 2 Arrays

```
#include<stdio.h>
#include<conio.h>

void main()

{     int a[10],b[10],i,j,m,n;
clrscr();
printf("Enter size of first array: ");
scanf("%d",&m);
printf("\nEnter %d elements into first array:\n",m);
for(i=0;i<m;i++)
scanf("%d",&a[i]);
printf("\nEnter the size of second array: ");
scanf("%d",&n);
printf("\nEnter %d elements into second array:\n",n);
for(i=0;i<n;i++)
scanf("%d",&b[i]);
for(i=m,j=0;j<n;i++,j++)
a[i]=b[j];
printf("\nFirst array after appending second array:\n");
for(i=0;i<m+n;i++)
printf("\n%d",a[i]);
getch();    }
```

6 Search in linear srch

main()

1:print "enter the size"
2:read n
3:print "enter n element into the array"
4:i=0
5:repeat steps 6 to 7 while i<n
6:read a[i] 7:i=i=1
8:print "enter the element to be searched"
9:read x 10:loc=search (a,n,x)
11:if (loc=-1)print "element is not found"
else print"element found at location",loc+1 12:end

search(a,n,x)

1:i=0
2:repeat steps 3 while i<n and a[i] x
3:i=1+1
4:if i=n return -1
else return

6 Search in lnear search

```
#include<stdio.h>
#include<conio.h>

int search(int [],int,int);
void main()
{   int a[10],i,n,x,loc;
clrscr();
printf("Enter the size : ");
scanf("%d",&n);
printf("\nEnter %d elements into the array:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nEnter the element to be searched : ");
scanf("%d",&x);
loc=search(a,n,x);
if(loc==-1) printf("\n%d is not found.",x);
else printf("\n%d is found at position %d in the array.",x,loc+1);
getch();   }

int search(int a[],int n,int v)
{   int i;
for(i=0;i<n && a[i]!=v;i++)
;
return i==n?-1:i;
}{  strcpy(temp,str[j]);
strcpy(str[j],str[j+1]);
strcpy(str[j+1],temp);
swapped=1;  } } }
```

7 Search in Bnry Srch

```
1:print "enter the size"
2:read n
3:print "enter n,"elements into the array in ascending order"
4:i=0
5:repeat steps 6 to 7 while i<n
6:read a[i]        7:i=i+1
8:print "enter the element searched;"
9:read x           10:loc=search (a,n,x)
11:if loc=-1,print "element is not found"
else print"element is not found at location",loc+1
12:end

search(a,n,x)
1:lb=0
2:ub=n-1
3:repeat steps 4 to 6 while lb<=ub
4:middle=(lb+ub)/2
5:if a [middle]=x return middle
6:if x<a[middle] ub=middle-1
  else lb=middle+1
7:return-1
```

7 Search in bnry srch

```
#include<stdio.h>
#include<conio.h>
int search(int [],int,int);

void main()
{   int a[10],i,n,x,loc;
clrscr();
printf("Enter the size : ");
scanf("%d",&n);
printf("\nEnter %d elements into the array in ascending order:\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nEnter the element to be searched : ");
scanf("%d",&x);
loc=search(a,n,x);
if(loc==-1) printf("\n%d is not found.",x);
else printf("\n%d is found at position %d in the array.",x,loc+1);
getch();   }

int search(int a[],int n,int v)
{   int lb,ub,middle;
lb=0;
ub=n-1;
while(lb<=ub)
{   middle=(lb+ub)/2;
if(a[middle]==v) return middle;
if(v<a[middle]) ub=middle-1;
else lb=middle+1;   }
return -1;   }
```

8 Sparse Matrix

```
1:start
2:declare structure struct{int r,c,v} s[10]
3:n=0
4:print "enter the dimensions"
5:read rows,cols       6:print "enter",rows*cols,"elements into the sparse matrix"
7:i=0
8:repeat steps 9 to 14 while i<rows
9:j=0
10:repeat steps 11 to 13 while j<cols
11:read x
12:if x!=0
  s[n].r=i
  s[n].c=j
  s[n].v=x;
  n=n+1
13:j=j+1            14:i=i+1
15:print "details of non zero elements"
  print "row coloumn values"
16:i=0
17:repeat steps 18 to 19 while i<n
18:print s[i].r s[i].c s[i].v
19:i=i+1            20:stop
```

8 Sparse Matrix

```
#include<stdio.h>
#include<conio.h>
void main()
{   struct   { int r,c,v; }s[10];
int rows,cols,n=0,i,j,x;
clrscr();
printf("Enter the dimension : ");
scanf("%d%d",&rows,&cols);
printf("\nEnter %d elements into the sparse matrix :\n",rows*cols);
for(i=0;i<rows;i++)
for(j=0;j<cols;j++)
{ scanf("%d",&x);
if(x!=0)
{ s[n].r=i;
s[n].c=j;
s[n].v=x;
n++;  } }
printf("\nDetails of non-zero elements in the sparse matrix:");
printf("\nRow\tColumn\tValue\n");
for(i=0;i<n;i++)
printf("\n%d\t%d\t%d",s[i].r,s[i].c,s[i].v);
getch();  }
```

9 Singly Linked List

```
#include<stdio.h>
#include<alloc.h>
#include<conio.h>
typedef struct listnode* listpointer;
struct listnode
{  int data;
listpointer next;  };

void main()
{  int i,n;
listpointer fnode,p;
clrscr();
printf("\nEnter the number of elements : ");
scanf("%d",&n);
if(n<1) return;
printf("\nEnter %d elements to list :\n",n);
fnode=(listpointer)malloc(sizeof(*fnode));
scanf("%d",&fnode->data);
fnode->next=NULL;
p=fnode;
for(i=0;i<n-1;i++)
{  p->next=(listpointer)malloc(sizeof(*p));
p=p->next;
scanf("%d",&p->data);  }
p->next=NULL;
printf("\nElements in the list :\n\n");
for(p=fnode;p!=NULL;p=p->next)
printf(" %d",p->data);
getch();  }
```

9 Singly linked list

```
1:start
2:define struct listnode with members data of type int and text next of type pointer
3:read number of elements of n
4:if(n<1)return
5:allocate space for one node and return adress if the node to pointer f node
6:read fnode->data
7:set fnode ->next=null
8:p=fnode      9:i=0
10:repeat steps 11 to 14 while i<n-1
11:allocate space for one node and return the adress of the node to pointer p
12:set p=p->next   13:read p->data
14:i=i+1       15:set p->next=null
16:set p=f node
17:repeat steps 18,19 while p not equal to null
18:print p->data     19:p=p->next     20:stop
```

13 doubly linked list:

```
1: start .
2: Define struct list node with member data, next.
3: Read number of elements members into n
4: if (n<1) return
5: Allocate space for one node and return address of the node Pointer f node.
6: Read fnode-> data.
7 set frode->next = Fnode -> prev = NULL
8: p= Frede.     9: initialize i=o
10: repeat steps 11-14 while i<n-1
11 Allocate space fora one node and return the address of the node to pointer p
12: set p -> next → prev = p, p=p->next.
13 read p-> data.     14 i=i+1
15 set p->next ->NULL, & node=p
16: set p = fnode.
17 Repeat steps 18,19while p not equal to null
18 point p->data
19: P=p->next      20: set p = node.
21: repeat step 15-19 while p not equal to null
22: print p->data      23: p=p-> prev.
24 stop.
```

13 Doubly linked list

```
#include<stdio.h>
#include<alloc.h>
#include<conio.h>
typedef struct listnode* listpointer;
struct listnode
{  int data;
listpointer next,prev;  };
void main()
{  int i,n;
listpointer fnode,lnode,p;
clrscr();
printf("\nEnter the number of elements : ");
scanf("%d",&n);
if(n<1) return;
printf("\nEnter %d elements into list :\n",n);
fnode=(listpointer)malloc(sizeof(*fnode));
scanf("%d",&fnode->data);
fnode->next=fnode->prev=NULL;
p=fnode;
for(i=0;i<n-1;i++)
{  p->next=(listpointer)malloc(sizeof(*p));
p->next->prev=p;
p=p->next;
scanf("%d",&p->data);  }
p->next=NULL;
lnode=p;
printf("\nListing elements forward :\n");
for(p=fnode;p!=NULL;p=p->next)
printf(" %d",p->data);
printf("\nListing elements backward :\n");
for(p=lnode;p!=NULL;p=p->prev)
printf(" %d",p->data);
getch();  }
```

14 Polynomial Additin

```
#include<stdio.h> #include<conio.h>
typedef struct {int coef,exp;} polynomial; void read(polynomial*,int);
void print(polynomial*,int);

void main()
{     polynomial p1[10],p2[10],p3[20]; int m,n,i,j,k;
clrscr();
printf("Number of terms in first polynomial : ");
scanf("%d",&m);
printf("\nEnter %d terms for first polynomial:\n",m);
read(p1,m);
printf("\nNumber of terms in second polynomial : ");
scanf("%d",&n);
printf("\nEnter %d terms for second polynomial:\n",n);
read(p2,n);
i=j=k=0; while(i<m && j<n)
{   if(p1[i].exp>p2[j].exp) p3[k++]=p1[i++];
else if(p1[i].exp<p2[j].exp) p3[k++]=p2[j++];
else if(p1[i].coef+p2[j].coef!=0)
{   p3[k]=p1[i++];
p3[k++].coef+=p2[j++].coef;   }
else i++,j++;  }
while(i<m) p3[k++]=p1[i++]; while(j<n)
p3[k++]=p2[j++]; printf("\nFirst polynomial : ");
print(p1,m);
printf("\nSecond polynomial : "); print(p2,n);
printf("\nPolynomial sum              : "); print(p3,k);
getch();  }

void read(polynomial p[],int n)
{  int i; for(i=0;i<n;i++)
{  printf("\nCoefficient : "); scanf("%d",&p[i].coef);
printf("Exponent          : "); scanf("%d",&p[i].exp);   }
}

void print(polynomial p[],int n)
{   int i;
if(p[0].exp==0) printf("%d",p[0].coef);
else if(p[0].exp==1) printf("%dx",p[0].coef);
else printf("%dx^%d",p[0].coef,p[0].exp);
for(i=1;i<n;i++)
{  if(p[i].coef>0) printf("+");
if(p[i].exp==0) printf("%d",p[i].coef);
else if(p[i].exp==1) printf("%dx",p[i].coef);
else printf("%dx^%d",p[i].coef,p[i].exp);   } }
```

15 SATACK ARRAY

```
#include<stdio.h> #include<conio.h>
#define LENGTH 5

int stack[LENGTH]; int top=0;
void push(int); void pop(); void list();

void main()
{               int ch,n; do
{               clrscr();
printf("\n\tSTACK OPERATIONS\n");
printf("\n\t\t1.PUSH"); printf("\n\t\t2.POP");
printf("\n\t\t3.LIST"); printf("\n\t\t4.EXIT");
printf("\n\n\tEnter your choice(1-4) : "); scanf("%d",&ch);
switch(ch)
{case 1:
printf("\nEnter the element : "); scanf("%d",&n);
push(n); break;
case 2:
pop(); break;
case 3:
list(); break;
default:continue;
}getch();  }while(ch!=4);   }

void push(int x)
{  if(top==LENGTH)
{  printf("\nSorry, stack is full..."); return;  }
stack[top++]=x;
printf("\n%d is added...",x);    }
void pop()
{  if(top==0)
{   printf("\nSorry, stack is empty..."); return;  }
printf("\n%d is removed...",stack[--top]);   }

void list() {
int i; if(top==0)
{  printf("\nSorry, stack is empty..."); return;  }
printf("\nThe stack :\n"); for(i=top-1;i>=0;i--)
printf("\n%d",stack[i]);   }
```

15 Stack Array Algebra

push(x)

1:start 2:If(top==LENGTH)
print "Overflow"
return
3:top=top+1 4:stack[top]=x
5:print x is added 6:return

pop() :-

1:start
2:If (top==Null)
print "underflow"
return
3:top=top-1 4:value=stack[top]
5:print value is removed 6:return

List() :-

1:start 2:if (top=0)
print "Underflow"
3:set i=top-1
4:repeat steps 5 to 6 while i>=0
5:print stack[i]
6:i=i-1 7:return

16 StaCK Linked List

```
#include<stdio.h> #include<conio.h> #include<alloc.h>
typedef struct stacknode* stackpointer; struct stacknode
{  int data; stackpointer next;   };
stackpointer top=NULL,node;
void push(int); void pop(); void list();
void main()
{               int ch,n; do
{               clrscr();
printf("\n\tSTACK OPERATIONS\n");
printf("\n\t\t1.PUSH"); printf("\n\t\t2.POP");
printf("\n\t\t3.LIST"); printf("\n\t\t4.EXIT");
printf("\n\n\tEnter your choice(1-4) : "); scanf("%d",&ch);
switch(ch)
{               case 1:
printf("\nEnter the element : "); scanf("%d",&n);
push(n); break;
case 2:
pop(); break;
case 3:
list(); break;
default:continue;          }
getch();
}while(ch!=4);             }
void push(int x)
{               node=(stackpointer)malloc(sizeof(*node));
if(node==NULL)
{               printf("\nSorry, insufficient memory...");
return;        }
node->data=x; node->next=top; top=node;
printf("\n%d is added...",x);              }
void pop()
{               if(top==NULL)
{               printf("\nSorry, stack is empty..."); return;
                }
printf("\n%d is removed...",top->data); node=top;
top=top->next; free(node);  }
void list()
{   if(top==NULL)
{    printf("\nSorry, stack is empty..."); return;    }
printf("\nThe stack :\n");
for(node=top;node!=NULL;node=node->next)
printf("\n%d",node->data);   }
```

17 Post Fix

```
#include<stdio.h> #include<conio.h>
int evaluate(char*);
void main()
{    char exp[15]; clrscr();
printf("Enter an expression in postfix form : "); gets(exp);
printf("\nResult = %d",evaluate(exp)); getch();   }
int evaluate(char *str)
{   int stack[15],top=0,op1,op2,i; for(i=0;str[i]!='\0';i++)
{   if(str[i]>='0' && str[i]<='9')
stack[top++]=str[i]-'0';
else
{  op2=stack[--top]; op1=stack[--top]; switch(str[i])
{ case '+':
stack[top++]=op1+op2; break;
case '-':
stack[top++]=op1-op2; break;
case '*':
stack[top++]=op1*op2; break;
case '/':
stack[top++]=op1/op2; break;
default:top+=2;  }  } }  return stack[--top];  }
```

17 PostFix Algebra

main() :-

1:start 2:Read exp
3:result=evaluate(exp)
4:print "Result=",result 5:stop

Evaluate(*str) :

1:start 2:top=0 3:i=0
4:Repeat steps 5 to 7 while str[i] ≠ NULL
5:If (str[i]>= '0' and str[i]<= '9')
stack[top]=str[i]='0' top=top+1
goto step 7 else top=top-1
op2=stack[top] top=top-1
op1=stack[top]
6:If (str[i] == '+') stack[top]=op1+op2
top=top+1 else if(str[i]== '-')
stack[top]=op1-op2 top=top+1
else if(str[i] == '*') stack[top]=op1*op2
top=top+1 else if(str[i]== '/')
stack[top]= op1/op2 top=top+1
else top=top+2 7:i=i+1
8:top=top-1 9:return stack[top]

14 Poly ADD: Algebra

1: Start
2: Declare struct {int coef, exp} polynomial

Main ():
1: start-
2 Read m
3 call read(p1,m)
4 Read n
5 call read (p2,n)
6 i =j=k= 0
7 Repeat steps 8 while i<m&j<n
8: if (p1 [i]. exp> p2 [j]. exp)
k = k+ 1
i = i+ 1
P3[K] = P 1[i]
else if (p) [i]. exp <p2[j]. exp)
k = k + 1
i = i + 1
P3[k] = p2[i]
else if (p1[i].coef +p2[j]. coef!=0)
i=i+1
P3[k]=p1[i]
k=k+1
j=j+1
P3[i]. coef = p3[k]. coef+p2[j]coef
else
i=i+1
j=j+1
9 Repeat step10 to 12 while i<m
10 k = k+ 1
11 i = i+1
12 P3[k]=p1[i]
13 Repeat steps 14 to 16 while j<n
14 = k= k+1 15: j=j+1
16: p3[k] = p1[j]
17 Call print (p1, m).
18 call point (p2,n)
19 call print (p3, k) 20 stop

read (p[ ], n):
1 start 2 i=0
3: Repeat steps 4 to 6 while i<n.
4: Read p[i]. coef
5: Real p[i].exp
6 i=i+1 7 return.

Print (P[ ], n):
1- Start-
2 IF (P[0]. exp== 0)
print p[0].coef,
else if (p[o]. exp==1) print p[0].coef, "x"
else point P[0]. coef , "x^"; p[0]. exp.
3 i= 1
4 Repeat steps 5 to 7 while i<n.
5 if (p[i].coet > 0) point "+"
6 if (prl[i]. exp==0) print P[i].coef
else if (P [i]. exp== 1) print p[i].coef,"x"
else print p[i].coef, "x^", p[i]. exp.
7 i=i+1. 8: return.

18 QUEUE ARRAY

```
#include<stdio.h> #include<conio.h>
#define LENGTH 5
int queue[LENGTH]; int front=0,rear=0;
void add(int); void del(); void list();
void main()
{   int ch,n; do
{   clrscr();
printf("\n\tQUEUE OPERATIONS\n");
printf("\n\t\t1.ADD"); printf("\n\t\t2.DELETE");
printf("\n\t\t3.LIST"); printf("\n\t\t4.EXIT");
printf("\n\n\tEnter your choice(1-4) : "); scanf("%d",&ch);
switch(ch)
{   case 1:
printf("\nEnter the element : "); scanf("%d",&n);
add(n); break;
case 2:    del(); break;
case 3:    list(); break;
default:continue;   }
getch();
}while(ch!=4);   }
void add(int x)
{   int i;
if(rear==LENGTH && front==0)
{   printf("\nSorry, queue is full..."); return;   }
if(rear==LENGTH)
{   for(i=front;i<rear;i++) queue[i-front]=queue[i];
rear=rear-front; front=0;   }
queue[rear++]=x; printf("\n%d is added...",x);   }
void del()
{   if(front==rear)
{  printf("\nSorry, queue is empty..."); return;  }
printf("\n%d is removed...",queue[front++]);   }
void list()
{   int i; if(front==rear)
{   printf("\nSorry, queue is empty..."); return;   }
printf("\nThe queue :\n\n"); for(i=front;i<rear;i++)
printf("%d\t",queue[i]);   }
```

18 QUEUE ARRAY Algebra

add(x) :

```
1:start
2:if (rear=LENGTH and front=NULL)
  print "overflow"
  return
3:if (rear ≠ LENGTH) goto step 10
4:i=front 5:repeat step 6,7 while i<rear
6:queue [i-front] = queue[i] 7:i=i+1
8:rear=rear-front 9:front=0
10:rear=rear+1  11:queue[rear]=x
12:print x is added        13:return
del()
1:start
2:if(front=rear)
  print "underflow"
  return
3:value=queue[front]. 4:front=front+1
5:print "value is removed"      6:return
list()
1:start
2:if (front==rear)
  print "underflow"
  return
3:i=front. 4:repeat steps 5,6 while i<rear
5:print queue[i]  6:i=i+1          7:return
```

19 QUEUE Linked Lista

```
#include<stdio.h> #include<conio.h> #include<alloc.h>
typedef struct queuenode* queuepointer; struct
queuenode
{   int data; queuepointer next;   };
queuepointer front=NULL,rear=NULL,node;
void add(int); void del(); void list();
void main()
{   int ch,n; do
{   clrscr();
printf("\n\tQUEUE OPERATIONS\n");
printf("\n\t\t1.ADD"); printf("\n\t\t2.DELETE");
printf("\n\t\t3.LIST"); printf("\n\t\t4.EXIT");
printf("\n\n\tEnter your choice(1-4) : "); scanf("%d",&ch);
switch(ch)
{   case 1:
printf("\nEnter the element : "); scanf("%d",&n);
add(n); break;
case 2:     del(); break;
case 3:     list(); break;   default:continue;  }
getch();
}while(ch!=4);   }
void add(int x)
{   node=(queuepointer)malloc(sizeof(*node));
if(node==NULL)
{   printf("\nSorry, insufficient memory..."); return;   }
node->data=x; node->next=NULL;
if(front==NULL) front=rear=node; else
{   rear->next=node; rear=rear->next;   }
printf("\n%d is added...",x);   }
void del()
{   if(front==NULL)
{  printf("\nSorry, queue is empty..."); return;  }
printf("\n%d is removed...",front->data); node=front;
front=front->next; free(node);   }
void list()
{  if(front==NULL)
{   printf("\nSorry, queue is empty..."); return;  }
printf("\nThe queue :\n\n");
for(node=front;node!=NULL;node=node->next)
printf("%d\t",node->data);   }
```

19 QUEUE Linked List Algorithm

```
1:start
2:Declare structure queuenode {int data;queue pointer
next*;}
add(x) :
1:start
2:allocate memory for one node and assign the address of
the node to pointer node
3:If (node == NULL)
  print "sorry,insufficientmemory..."
  return
4:node ->data =x      5:Node ->next=NULL
6:If(front== NULL)
  front=rear=node
  else
  rear ->next=node
  rear=rear ->next
7:Print x "is added..."       8:return
del():
1:start
2:If (front==NULL)
  Print "sorry,queue is empty..."
  return
3:Print front ->data is removed
4:node=front  5:front= front ->next
6: free(node)            7: return
list():
1:start
2:If (front == NULL)
  print "sorry,queue is empty..."
  return
3:node=front
4:Repeat step 5 to 6 while node ≠ NULL
5:print node ->data
6:node=node -> next          7:return
```

20 PREORDER

```
#include<stdio.h> #include<conio.h> #include<alloc.h>
typedef struct treeNode* treePointer; struct treeNode
{  int data;
treePointer leftChild,rightChild;  };
treePointer root=NULL;
void create();
void preOrder(treePointer);
void main()
{   int choice; do
{   clrscr();
printf("\n\tBINARY SEARCH TREE OPERATIONS\n");
printf("\n\t\t1. Create"); printf("\n\t\t2. Traverse
preorder"); printf("\n\t\t3. Exit"); printf("\n\n\tEnter your
choice(1-3) : "); scanf("%d",&choice);
switch(choice)
{  case 1:
create(); break;
case 2:
if(root==NULL) printf("\nTree is empty..."); else
{   printf("\nPreorder traversal is "); preOrder(root);  }
break; default:continue;   }
getch();
}while(choice!=3);  }
void create()
{   int i,n=0,left; treePointer node,temp,p; if(root!=NULL)
{   printf("\nTree already exists..."); return;   }
printf("\nEnter the number of nodes : ");
scanf("%d",&n);
printf("\nEnter %d nodes for the binary search
tree:\n",n); root=(treePointer)malloc(sizeof(*root));
scanf("%d",&root->data);
root->leftChild=root->rightChild=NULL; for(i=0;i<n-1;i++)
{   temp=(treePointer)malloc(sizeof(*temp));
scanf("%d",&temp->data);
temp->leftChild=temp->rightChild=NULL; node=root;
while(node!=NULL)
{   left=0; p=node;
if(temp->data<node->data)
{   left=1;
node=node->leftChild;   }
else node=node->rightChild;   }
if(left) p->leftChild=temp; else p->rightChild=temp;  }
printf("\n\nBinary search tree created with %d
nodes...",n);  }
void preOrder(treePointer node)
{  if(node==NULL) return; printf(" %d",node->data);
preOrder(node->leftChild); preOrder(node->rightChild);  }
```

25 SORT SELECTION

```
main():-
1. start        2.read n        3. i=0
4. repeat steps 5,6 while i<n
5. read a[i]. 6. i=i+1          7. i=0
8 repeat steps 9,10 while i<n>
9. print a[i]  10. i=i+1
11. call sort[a,n]  12. i=0
13. repeat steps 14,15 while i<n>
14. print a[i]  15. i=i+1         16. stop
sort (a,n):-
1. start            2. i=0
3. repeat steps 4 to 13 i<n-1
4. index =i         5. j=i+1
6. repeat steps 7,8 i<n>
7. if(a[i]<a[index]) index=j;
8. j=j+1
9. if(index==i)continue
10. temp=a[i]
11. a[i]=a[index]12. a[index]=temp
13. i=i+1        14. return
```

25 SORT SELECTION

```
#include<stdio.h>
#include<conio.h>
void sort(int*,int); void main()
{  int a[10],n,i;
clrscr();
printf("Enter the size : "); scanf("%d",&n);
printf("\nEnter %d elements into the array :\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nArray before sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); sort(a,n);
printf("\n\nArray after sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); getch();   }

void sort(int a[],int n)
{   int i,j,index,temp; for(i=0;i<n-1;i++)
{  index=i; for(j=i+1;j<n;j++) if(a[j]<a[index]) index=j;
if(index==i) continue; temp=a[i];
a[i]=a[index]; a[index]=temp;   }  }
```

20 PREORDER ALGEBRA

1.start
2.declare structure treenode{int data;treepointer leftchild,rightchild;}
3.treepointer root=null
create():=
1.start 2.n=0
3.if(root!=null)
3.1 print "tree already exist...."
3.2 return 4.read n
5.allocate memory for one node and assign the adress of the node to pointer root
6.read root->data
7.root->leftchild=root->rightchild=null 8.i=0
9.repeat steps 10 to 19 while i<n-1
10.allocate memory for one node and assign the address of the node to pointer temp
11.read temp->data
12.temp->leftchild=temp->rightchild=null
13.node=root
14.repeat steps 15 to 17 while node!=null
15.left=0 16.p=node
17.if(temp->data=node->data)
left=1
node=node->leftchild
else
node=node->rigtchild
18.if(left) p->leftchild=temp
else p->rightchild=temp
19.i=i+1
20.print "binary search tree created with",n,"node..."
21.return
preorder(root):-
1.start 2.if(node==null) return 3.print root->data
4.call preoder(node->leftchild)
5.call preoder(node->righttchild) 6.return

16 Stack linked List ALGEBRA

1:start
2:Define structure with members data of type int and next of type stackpointer
push(x)
1:start
2:Allocate space for one node and return the address of the node to pointer node
3:if(node=NULL)
print "insufficient memory"
return
4:node ->data=x
5:node ->next=top 6:top=node
7:print x is added 8:return
pop() :-
1:start
2:If (top==Null)
print "underflow"
return
3:print top->data is removed
4:node=top 5:top=top ->next
6:free(node) 7:return
List() :-
1:start
2:if (top=Null)
print "Underflow"
return
3:node=top
4:repeat steps 5 to 6 while (node ≠ NULL)
5:print node ->data
6:node=node ->next 7:return

21 BST INORDER

```
#include<stdio.h> #include<conio.h> #include<alloc.h>
typedef struct treeNode* treePointer; struct treeNode
{  int data;
treePointer leftChild,rightChild;  };
treePointer root=NULL;
void create();
void inOrder(treePointer);
void main()
{   int choice; do
{   clrscr();
printf("\n\tBINARY SEARCH TREE OPERATIONS\n");
printf("\n\t\t1. Create"); printf("\n\t\t2. Traverse
inorder"); printf("\n\t\t3. Exit");
printf("\n\n\tEnter your choice(1-3) : ");
scanf("%d",&choice);
switch(choice)
{ case 1:
create(); break;
case 2:
if(root==NULL) printf("\nTree is empty..."); else
{  printf("\nInorder traversal is "); inOrder(root);  }
break; default:continue; }
getch();   }while(choice!=3);   }
void create()
{ int i,n=0,left; treePointer node,temp,p; if(root!=NULL)
{  printf("\nTree already exists..."); return;  }
printf("\nEnter the number of nodes : ");
scanf("%d",&n);
printf("\nEnter %d nodes for the binary search
tree:\n",n); root=(treePointer)malloc(sizeof(*root));
scanf("%d",&root->data);
root->leftChild=root->rightChild=NULL; for(i=0;i<n-1;i++)
{  temp=(treePointer)malloc(sizeof(*temp));
scanf("%d",&temp->data);
temp->leftChild=temp->rightChild=NULL; node=root;
while(node!=NULL)
{  left=0; p=node;
if(temp->data<node->data)
{  left=1;
node=node->leftChild;}
else node=node->rightChild;}
if(left) p->leftChild=temp; else p->rightChild=temp;}
printf("\n\nBinary search tree created with %d
nodes...",n);   }
void inOrder(treePointer node)
{  if(node==NULL) return; inOrder(node->leftChild);
 printf(" %d",node->data); inOrder(node->rightChild);   }
```

21 BST INORDER ALGRTHM

1.start
2.declare structure treenode{int dat;TREEPOINTER leftchild,rightchild;}
3.treepointer root=NULL;
create()
1.start 2.n=0
3.if(root!=null)
3.1 print("tree already exist...")
3.2 return
4.read n
5.allocate memory for one and assign the address of the node to pointer root
6.read root->data
7.root->leftchild=root->rightchild=null.
8.i=0 9.repeat step 10 to 19 whilw i<n-1
10.allocate memory for one node and assign to address of the node to pointer temp
11.read temp->data
12.temp->leftchild=temp->rightchild=null
13.node=root
14.repeat step 15 to 17 while node!=null
15.left=0 16.p=node
17.if(temp->data=node->data)
left=1
node=node->leftchild
else
node=node->rightchild
18.if(left) p->leftchild=temp
else p->rightchild=temp
19.i=i+1
20.print "binary search tree created with"n"node"
21.return

inorder(root);-
1.start 2.if(node==null)return;
3.call inorder(node->leftchild) 4.print node->data
5.call inorder(node->rightchild) 6.return

24 SORT EXCHANGE

```
#include<stdio.h>    #include<conio.h>
void sort(int*,int);
void main()
{  int a[10],n,i; clrscr();
printf("Enter the size : "); scanf("%d",&n);
printf("\nEnter %d elements into the array :\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nArray before sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); sort(a,n);
printf("\n\nArray after sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); getch();  }
void sort(int a[],int n)
{    int i,j,temp; for(i=0;i<n-1;i++)
{  for(j=i+1;j<n;j++) if(a[i]>a[j])
{    temp=a[i]; a[i]=a[j]; a[j]=temp;   } }  }
```

24 SORT EXCHANGE ALGEBRA

main():-
1.start 2.read n 3.i=0
4.repeat steps 5,6 while i<n
5.read a[i] 6.i=i+1
8.repeat steps 9,10 while i<n
9.print a[i] 10.i=i+1
11.call sort9a,n) 12.i=0
13.repeat step 14,15 while i<n
14.print a[i]
15.i=i+1 16.stop
sort(a,n):-
1.start 2.i=0
3.repeat steps 4 to 8 while i<n-1
4.j=j+1 5.repeat steps 6,7 while j<n
6.if(a[i]>a[j])
6.1 temp=a[i]
6.2 a[i]=a[j]
6.3 a[j]=temp
7.j=j+1 8.i=i+1 9.return

26 SORT INSERTION

```
#include<stdio.h> #include<conio.h>
void sort(int*,int);
void main()
{  int a[10],n,i; clrscr();
printf("Enter the size : "); scanf("%d",&n);
printf("\nEnter %d elements into the array :\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nArray before sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); sort(a,n);
printf("\n\nArray after sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); getch();  }
```

26 SORT INSERTION ALGEBRA

main():-
1.start 2.read n
3.i=0 4.repeat step 5,6 while i<n
5.read a[i] 6.i=i+1 7.i=0
8.repeat steps 9,10 while i<n
9.print a[i] 10.i=i+1 11.call sort(a,n)
12.i=0 13.repeat steps 14,15 while i<n
14.print a[i] 15.i=i+1 16.stop
sort(a,n):-
1.start 2.i=1
3.repeat step 4 to 10 while i<n
4.x=a[i] 5.j=i-1
6.repeat steps 7,8 while x<a[j] & j>=0
7.a[j+1]=a[j] 8.j=j-1 9.a[j+1]=x
10.i=i+1 11.return

27 QUICK SORT ALG

main():-

1. start 2.read n 3. i=0
4. repeat steps 5,6 while i<n
5. read a[i] 6. i=i+1 7. i=0
8. repeat steps 9,10 while i<n
9. print a[i] 10. i=i+1
11. call sort (a,n) 12. i=0
13. repeat step14,15 while i<n
14. print a[i] 15 i=i+1 16 stop

sort(a,n):-

1. start 2.call qsort(a,o,n-1)
3. return

qsort(a[],lb,ub)

1. start 2. if(lb>=ub)return 3. c=a[lb]
4. ub=lb 5.down=ub
6. repeat steps 7 to14
7. repeat steps 8 while a[up]<c & upc down
8. up=up+1
9. repeat steps 10 while a[down]>c
10. down=down-1
11. if (up>=down)go to step 15
12. tump=a[up] 13. a[up]=a[down]
14.a[down]=temp; 15.a[lb]=a[down]
16.a[doewn]=c 17.call qsort(a,lb,down-1)
18.call qsort(a,down+1,ub) 19.return

27 QUICK SORT

```
#include<stdio.h> #include<conio.h>
void sort(int*,int); void qsort(int*,int,int);
void main()
{    int a[10],n,i; clrscr();
printf("Enter the size : "); scanf("%d",&n);
printf("\nEnter %d elements into the array :\n",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nArray before sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); sort(a,n);
printf("\n\nArray after sorting :"); for(i=0;i<n;i++)
printf(" %d",a[i]); getch(); }
void sort(int a[],int n)
{  qsort(a,0,n-1); }
void qsort(int a[],int lb,int ub)
{  int up,down,c,temp; if(lb>=ub) return; c=a[lb];
up=lb; down=ub; while(1)
{  while(a[up]<=c && up<down) up++;
while(a[down]>c) down--; if(up>=down) break;
temp=a[up]; a[up]=a[down]; a[down]=temp;  }
a[lb]=a[down]; a[down]=c; qsort(a,lb,down-1);
qsort(a,down+1,ub);  }
```

22 POST ORDER

```
#include<stdio.h> #include<conio.h> #include<alloc.h>
typedef struct treeNode* treePointer; struct treeNode
{  int data;
 treePointer leftChild,rightChild;  };
treePointer root=NULL;
void create();
void postOrder(treePointer);
 void main()
{  int choice; do
{  clrscr();
printf("\n\tBINARY SEARCH TREE OPERATIONS\n");
printf("\n\t\t1. Create"); printf("\n\t\t2. Traverse
postorder"); printf("\n\t\t3. Exit"); printf("\n\n\tEnter
your choice(1-3) : "); scanf("%d",&choice);
switch(choice)
{  case 1:
create(); break;
case 2:
if(root==NULL) printf("\nTree is empty..."); else
{printf("\nPostorder traversal is "); postOrder(root);  }
break;
default:continue; }
getch();
}while(choice!=3);  }
void create()
{int i,n=0,left;
treePointer node,temp,p;
if(root!=NULL)
{  printf("\nTree already exists...");
return;  }
printf("\nEnter the number of nodes : ");
scanf("%d",&n);
printf("\nEnter %d nodes for the binary search
tree:\n",n);
root=(treePointer)malloc(sizeof(*root));
scanf("%d",&root->data);
root->leftChild=root->rightChild=NULL;
for(i=0;i<n-1;i++)
{  temp=(treePointer)malloc(sizeof(*temp));
scanf("%d",&temp->data);
temp->leftChild=temp->rightChild=NULL;
node=root;
while(node!=NULL)
{  left=0;
p=node;
if(temp->data<node->data)
{  left=1;
node=node->leftChild;  }
else node=node->rightChild;  }
if(left) p->leftChild=temp;
else p->rightChild=temp; }
printf("\n\nBinary search tree created with %d
nodes...",n); }
void postOrder(treePointer node)
{ if(node==NULL) return; postOrder(node->leftChild);
postOrder(node->rightChild);
printf(" %d",node->data);  }
```

22 POST ORDER ALGRTHM

1 start
2 Define structure tree node with member data of type int and leftchild, rightchild of type treepointer

create()

1 start
2 if(not=null)
  print tree already exist
  return
3 read the number of nodes to n
4 allocate space for one node and return address of the node to the pointer not
5 read root->data
6 set root->leftchild,root->rightchild=null
7 set i=0
8 repeat step 9-15 while (i<n-1)
9 allocate space for one node and return address of the node to the pointer temp
10 read temp->data
11 set temp->leftchild,temp->rightchild=null
12 set node root
13 repeat step 14-15 while (node1=NULL)
14 set left=0
15 set p=node
16 if (temp->data <node->data)
    setleft=1
    node=node->leftchild
    else
    node=node->rightchild
17 if(left=1)
    p->leftchild=temp
    else
    p->rightchild=temp
18 i=i+1
19 print binary search tree created with a node
20 return

search(x)
1 set node=root
2 repeat step 3-4 while (node!=NULL)
3 if (node->data=x)
   print 'element is found'
4 if(x<node->data)
   node=node->leftchild
   else
   node=node->rightchild
5 'element not found'

23 BST SEAERCH

```
#include<stdio.h> #include<conio.h> #include<alloc.h>
typedef struct treeNode* treePointer; struct treeNode
{  int data;
treePointer leftChild,rightChild;  };
treePointer root=NULL;
void create();
void search(int);

void main()
{  int choice,n; do
{  clrscr();
printf("\n\tBINARY SEARCH TREE OPERATIONS\n");
printf("\n\t\t1. Create"); printf("\n\t\t2. Search");
printf("\n\t\t3. Exit"); printf("\n\n\tEnter your choice(1-3)
: "); scanf("%d",&choice);
switch(choice)
{  case 1:
create(); break;
case 2:
if(root==NULL) printf("\nTree is empty..."); else
{  printf("\nEnter the element : "); scanf("%d",&n);
search(n);  }
break; default:continue;  }
getch();
}while(choice!=3);  }

void create()
{  int i,n=0,left; treePointer node,temp,p; if(root!=NULL)
{  printf("\nTree already exists..."); return;  }

printf("\nEnter the number of nodes : "); scanf("%d",&n);
printf("\nEnter %d nodes for the binary search
tree:\n",n); root=(treePointer)malloc(sizeof(*root));
scanf("%d",&root->data);
root->leftChild=root->rightChild=NULL; for(i=0;i<n-1;i++)
{  temp=(treePointer)malloc(sizeof(*temp));
scanf("%d",&temp->data);
temp->leftChild=temp->rightChild=NULL; node=root;
while(node!=NULL)
{  left=0; p=node;
if(temp->data<node->data)
{  left=1;
node=node->leftChild;  }
else node=node->rightChild;  }
if(left) p->leftChild=temp; else p->rightChild=temp;  }
printf("\n\nBinary search tree created with %d
nodes...",n);  }
void search(int x)
{  treePointer node; node=root; while(node!=NULL)
{  if(node->data==x)
{  printf("\nElement is found"); return;  }
if(x<node->data) node=node->leftChild; else node=node-
>rightChild;  }
printf("\nElement is not found");  }
```

23 BST SEAERCH ALG

1 start
2 Define structure tree node with member data of type int and leftchild, rightchild of type treepointer

create()

1 start
2 if(not=null)
  print tree already exist
  return
3 read the number of nodes to n
4 allocate space for one node and return address of the node to the pointer not
5 read root->data
6 set root->leftchild,root->rightchild=null
7 set i=0
8 repeat step 9-15 while (i<n-1)
9 allocate space for one node and return address of the node to the pointer temp
10 read temp->data
11 set temp->leftchild,temp->rightchild=null
12 set node root
13 repeat step 14-15 while (node1=NULL)
14 set left=0
15 set p=node
16 if (temp->data <node->data)
    setleft=1
    node=node->leftchild
    else
    node=node->rightchild
17 if(left=1)
    p->leftchild=temp
    else
    p->rightchild=temp
18 i=i+1
19 print binary search tree created with a node
20 return

search(x)

1 set node=root
2 repeat step 3-4 while (node!=NULL)
3 if (node->data=x)
   print 'element is found'
4 if(x<node->data)
   node=node->leftchild
   else
   node=node->rightchild
5 'element not found'