

## and Algorithms

### - Programs -

- Suresh -

1. Write a program to maintain a stack using array.

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

typedef struct
{
    int a[100];
    int top;
} stack;

void push( stack *s, int x)
{
    if (s->top == 99) printf("Stack Full");
    else
        s->a[++s->top] = x;
}
```

- Suresh  
9322264001

```
int pop ( stack *s )
{
    int x;
    if ( s->top < 0 )
        { printf("stack empty"),
          return -1;
        }
    else
        { x = s->a [ s->top-- ];
          return x;
        }
}

void display ( stack s )
{
    int i;
    for ( i = s.top; i >= 0; --i )
        printf("%d\n", s.a[i]);
}

void main()
{
    stack s1;
    int x, c;
    s1.top = -1; /* stack initialization */
}
```

```
do { clrscr();
      printf(" MENU\n");
      printf(" 1 - PUSH\n");
      printf(" 2 - POP\n");
      printf(" 3 - DISPLAY\n");
      printf(" 4 - EXIT\n");
      printf(" Enter your choice");
      scanf("%d", &c);
      switch(c)
      {
        case 1: scanf("%d", &x);
                   push(&s1, x);
                   break;
        case 2: scanf("%d", &x);
                   x = pop(&s1);
                   if(x != 1) printf("%d", x);
                   break;
        case 3: display(s1);
        }
        getch();
      }
    while(c != 4);
    getch();
} /* end of main */
Note: #include <conio.h>, clrscr(), getch() are
optional.
```

-Sureshan  
9322264001

- 4 -

write a program to maintain two stacks  
in one array:

```
#include <stdio.h>
#include <conio.h>
void push1( int a[], int *top1, int top2, int x)
{
    if (*top1 + 1 == top2) printf("FULL");
    else {
        *top1 = *top1 + 1;
        a[*top1] = x;
    }
}
void push2( int a[], int *top2, int top1, int x)
{
    if (*top2 - 1 == top1) printf("FULL");
    else {
        *top2 = *top2 - 1;
        a[*top2] = x;
    }
}
int pop1( int a[], int *top1)
{
    int x;
    if (*top1 < 0) { printf("EMPTY");
                      return -1;
    }
    else { x = a[*top1];
            *top1 = *top1 - 1;
            return x;
    }
}
```

```
int pop2( int a[], int *top2)
{
    int x;
    if (*top2 == 100) { printf("EMPTY");
                        return -1;
    }

    else { x = a[*top2];
            *top2 = *top2 + 1;
            return x;
    }
}
```

```
void display1( int a[], int top1)
{
    int i;
    for(i = top1; i >= 0; --i)
        printf("%d\n", a[i]);
}

void display2( int a[], int top2)
{
    int i;
    for(i = top2; i < n; ++i)
        printf("%d\n", a[i]);
}
```

- Survehan  
9322264001

```
void main()
{
    int a[100], top1=-1, top2=100;
    int x, c;

    do {
        clrscr();
        printf("      MENU\n");
        printf(" 1      PUSH1\n");
        printf(" 2      PUSH2\n");
        printf(" 3      POP1\n");
        printf(" 4      POP2\n");
        printf(" 5      DISPLAY1\n");
        printf(" 6      DISPLAY2\n");
        printf(" 7      EXIT\n");
        printf(" Enter your choice");
        scanf("%d", &c);
        switch(c)
        {
            case 1: scanf("%d", &x);
                      push1(a, &top1, top2, x);
                      break;
            case 2: scanf("%d", &x);
                      push2(a, &top2, top1, x);
                      break;
            case 3: x = pop1(a, &top1);
                      if(x != -1) printf("%d", x);
                      break;
            case 4: x = pop2(a, &top2);
                      if(x != -1) printf("%d", x);
                      break;
        }
    } while(c != 7);
}
```

```
case 5: display(a, top1);
break;

case 6 : display2(a, top2);
break;

} /* end of switch */
getch();
}
while( c != '#');
getch();
} /* end of main */
```

3. write a program to maintain a stack using linked list:

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

/* linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
} NODE;

/* datatype */
typedef struct
{
    NODE *top;
} stack;
```

Sureshan  
9322264001

- 8 -

```
void push( stack *s, int x)
{
    NODE *ptr;
    ptr = (NODE*) malloc(sizeof(NODE));
    ptr->data = x;
    ptr->next = s->top;
    s->top = ptr;
}

int pop( stack *s)
{
    int x;
    NODE *ptr;

    if (s->top == NULL) {
        printf("EMPTY");
        return -1;
    }

    else {
        ptr = s->top;
        x = ptr->data;
        s->top = ptr->next;
        free(ptr);
        return x;
    }
}
```

Sureshwan  
9322264001

```
void display (stack s)
{
    NODE *ptr;
    ptr = s.top;
    while (ptr != NULL)
    {
        printf ("%d\n", ptr->data);
        ptr = ptr->next;
    }
}

void main()
{
    stack s1;
    int x, c;
    s1.top = NULL; /* stack initialization */
    do {
        clrscr();
        printf(" 1 - PUSH\n");
        printf(" 2 - POP\n");
        printf(" 3 - DISPLAY\n");
        printf(" 4 - EXIT\n");
        printf(" Enter your choice");
        scanf("%d", &c);
    } while (c != 4);
}
```

```
switch ( c )
{
    case 1: scanf( "%d", &x );
               push( &s1, x );
               break;

    case 2:   x = pop( &s1 );
               if (x != -1) printf( "%d", x );
               break;

    case 3:   display( s1 );

}

getch();
}

while ( c != 4 );
getch();

} /* end of main */
```

4) Write to maintain a double ended queue (dequeue) (11)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
/* Program to maintain double ended queue using linked list */

/* linked list node structure */

typedef struct node
{
    int data;
    struct node *left,*right;
}NODE;

/*type declaration of deque*/

typedef struct
{
    NODE * l,*r;
}dq;

void addleft(dq * q,int x)
{
    NODE * ptr;
    ptr=(NODE *)malloc(sizeof(NODE));
    ptr->data=x;
    ptr->left=ptr->right=NULL;
    if(q->l==NULL)
        q->l=q->r=ptr;
    else
        { ptr->right=q->l;
          q->l->left=ptr;
          q->l=ptr;
        }
}
void addright(dq * q,int x)
{
    NODE * ptr;
    ptr=(NODE *)malloc(sizeof(NODE));
    ptr->data=x;
    ptr->left=ptr->right=NULL;
    if(q->r==NULL)
        q->l=q->r=ptr;
```

```
else

else
    { ptr->left=q->r;
      q->r->right=ptr;
      q->r=ptr;
    }
}

int deleteleft(dq *q)
{
    NODE * ptr;
    int x;
    if(q->l==NULL){printf("Empty dq");
                      return -1;
                    }

    else
    {
        ptr=q->l;
        x=ptr->data;

        q->l=ptr->right;
        if(q->l==NULL) q->r=NULL;
        else
            q->l->left=NULL;
        free(ptr);
        return x;
    }
}

int deleteright(dq *q)
{
    NODE * ptr;
    int x;
    if(q->r==NULL){printf("Empty dq");
                      return -1;
                    }

    else
    {
        ptr=q->r;
        x=ptr->data;
```

```

        q->r=ptr->left;
        if(q->r==NULL) q->l=NULL;
        else
            q->r->right=NULL;
        free(ptr);
        return x;
    }
}

void display(dq q)
{
    NODE *p;
    p=q.l;
    if(p==NULL)printf("DQ IS EMPTY");
    else
        while(p!=NULL)
    {
        printf("%d ",p->data);
        p=p->right;
    }
}

void main()
{
    dq q1;
    int x,c;
    q1.l=q1.r=NULL;
    do
    { clrscr();
        printf("Enter your choice\n");
        printf("1      To add left\n");
        printf("2      To add right\n");
        printf("3      To delete from left\n");
        printf("4      To delete from right \n");

        printf("5      To display\n");
        printf("6      To quit\n");
        scanf("%d",&c);
    }
}

```

```
switch(c)
{
    case 1:printf("Enter the element to be inserted");
              scanf("%d",&x);
              addleft(&q1,x);
              printf("DQ AFTER ADDITION\n");
              display(q1);
              getch();
              break;
    case 2:printf("Enter the element to be inserted");
              scanf("%d",&x);
              addright(&q1,x);
              printf("DQ AFTER ADDITION\n");
              display(q1);
              getch();
              break;

    case 3:x=deleteleft(&q1);
              if(x>=0)printf("element deleted %d\n",x);
              getch();
              break;
    case 4:x=deleteright(&q1);
              if(x>=0)printf("element deleted %d\n",x);
              getch();
              break;

    case 5: printf("DQ is: ");
              display(q1);
              getch();
              break;
    default: exit(1);
}
}

while(c!=6);
```

5) wAP to concatenate two linked lists

(19)

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

/* program to concatenate two given linked lists */

/*linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
} NODE;

/*function to create a linked list of n integers */

NODE * create(int n)
{
    NODE * start=NULL,*ptr,*prev;
    int i;
    for(i=1;i<=n;++i)
    {
        ptr=(NODE *)malloc(sizeof(NODE));
        printf("Enter the data\n");
        scanf("%d",&ptr->data);
        ptr->next=NULL;
        if(start==NULL)start=ptr;
        else
            prev->next=ptr;
        prev=ptr;
    }
    return start;
}
void display(NODE * start)
{
    if(start==NULL)printf("NULL");
    else
        while(start!=NULL)
        {
            printf("%d ",start->data);
            start=start->next;
        }
        printf("\n\n");
}
```

(15)

```
void concatenate(NODE * * start1,NODE *start2)
{
    NODE * ptr;
    if(*start1==NULL)
        *start1=start2;
    else
    {
        ptr=*start1;
        while(ptr->next!=NULL)
            ptr=ptr->next;
        ptr->next=start2;
    }
}
void main()
{
    NODE *start1,*start2;
    int m,n;
    clrscr();
    printf("enter number of elements in the lists\n");
    scanf("%d%d",&n,&m);
    printf("\n creating first list\n\n");
    start1=create(n);
    printf("\n creating second list\n\n");
    start2=create(m);
    clrscr();
    printf("\nFIRST LIST: start1-> ");
    display(start1);
    printf("\nSECOND LIST: start2-> ");
    display(start2);

    concatenate(&start1,start2);
    printf("\nLIST AFTER CONCATENATION :start1-> ");
    display(start1);
    getch();
}
```

(16)

6) WAP to find the reverse of a given linked list.

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

/* program to find reverse of a given linked list */

/* linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
} NODE;

/*function to create a linked list of n integers */

NODE * create(int n)
{
    NODE * start=NULL,*ptr,*prev;
    int i;
    for(i=1;i<=n;++i)
    {
        ptr=(NODE *)malloc(sizeof(NODE));
        printf("Enter the data\n");
        scanf("%d",&ptr->data);
        ptr->next=NULL;
        if(start==NULL) start=ptr;
        else
            prev->next=ptr;
        prev=ptr;
    }
    return start;
}

void reverse(NODE ** start)
{
    NODE * ptr,*prev,*ptrl;
    ptr=*start;
    prev=NULL;
```

(17)

(18)

```
while(ptr!=NULL)
{
    ptr1=ptr->next;
    ptr->next=prev;
    prev=ptr;
    ptr=ptr1;
}
*start=prev;
}
void display(NODE * start)
{
    if(start==NULL)printf("NULL");
    else
        while(start!=NULL)
        {
            printf("%d ",start->data);
            start=start->next;
        }
    printf("\n\n");
}
void main()
{
    NODE *start1;
    int n;
    clrscr();
    printf("enter number of elements in the list\n");
    scanf("%d",&n);
    printf("\n creating list\n\n");
    start1=create(n);
    clrscr();
    printf("\nGIVENT LIST: start1->    ");
    display(start1);

    reverse(&start1);
    printf("\nREVERSE OF THE GIVEN LIST :start1->      ");
    display(start1);
    getch();
}
```

7) WAP to find the intersection of two given linked lists.  
(creating a linked list containing common elements of both lists)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
/* program to find intersection of two given linked lists */

/*linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
} NODE;

/* function for creating a linked list */

NODE * create(int n)
{
    NODE * start=NULL,*ptr,*prev;
    int i;
    for(i=1;i<=n;++i)
    {
        ptr=(NODE *)malloc(sizeof(NODE));
        printf("Enter the data\n");
        scanf("%d",&ptr->data);
        ptr->next=NULL;
        if(start==NULL) start=ptr;
        else
            prev->next=ptr;
        prev=ptr;
    }
    return start;
}

void disp(NODE * start)
{
    while(start!=NULL)
    {
        printf("%d ",start->data);
        start=start->next;
    }
    printf("\n\n");
}
```

```
void main()
{
    NODE *start1,*start2,*start3=NULL,*ptr,*ptr1,*p,*prev;
    int m,n;
    clrscr();
    printf("enter number of elements in the lists\n");
    scanf("%d%d",&n,&m);
    printf("\n creating first list\n\n");
    start1=create(n);
    printf("\n creating second list\n\n");
    start2=create(m);
    clrscr();
    ptr=start1;

    while(ptr!=NULL)
    {
        ptr1=start2;
        while(ptr1!=NULL)
        {
            if(ptr->data==ptr1->data){
                p=(NODE *)malloc(sizeof(NODE));
                p->data=ptr->data;
                p->next=NULL;
                if(start3==NULL) start3=p;
                else prev->next=p;
                prev=p;
            }
            ptr1=ptr1->next;
        }
        ptr=ptr->next;
    }
    printf("elements in first list    ");
    disp(start1);
    printf("elements in second list    ");
    disp(start2);
    printf("intersection list    ");
    disp(start3);
    getch();
}
```

8) wAP to delete an element from a given linked list.

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

/* program to delete a given element from a given linked list */

/*linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
} NODE;

/*function to create a linked list */

NODE * create(int n)
{
    NODE * start=NULL,*ptr,*prev;
    int i;
    for(i=1;i<=n;++i)
    {
        ptr=(NODE *)malloc(sizeof(NODE));
        printf("Enter the data\n");
        scanf("%d",&ptr->data);
        ptr->next=NULL;
        if(start==NULL)start=ptr;
        else
            prev->next=ptr;
        prev=ptr;
    }
    return start;
}
void deletion(NODE **start,int x)
{
    NODE * ptr,*prev;
    ptr=*start;
    prev=NULL;
    /* searching for x */
    while(ptr!=NULL)
        if(x==ptr->data)
            break;

```

```
else
    {prev=ptr;
    ptr=ptr->next;
    }

if(ptr==NULL)
    printf("Element is not present\n");
else
{
    if(prev==NULL)
        *start=ptr->next;
    else
        prev->next=ptr->next;
    free(ptr);
}
}

void display(NODE * start)
{
    if(start==NULL)printf("NULL");
    else
        while(start!=NULL)
        {
            printf("%d ",start->data);
            start=start->next;
        }
    printf("\n\n");
}

void main()
{
    NODE *start1;
    int n,x;
    clrscr();
    printf("enter number of elements in the list\n");
    scanf("%d",&n);
    printf("\n creating list\n\n");
    start1=create(n);
    clrscr();
```

```
printf("\nGIVENT LIST: start1->  ");
display(start1);
printf("enter element to be deleted\n");
scanf("%d",&x);
deletion(&start1,x);
printf("\nLIST AFTER DELETION  :start1->      ");
display(start1);
getch();
}


```

q) Maintaining a linked list of students according to roll no: in ascending order. (24)

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

/* program to maintain a linked list of students in ascending
order
according to roll number using a function to insert an element
into a linked list */

/* type declaration for student */

typedef struct
{
    char name[30];
    int roll;
}student;

/* linked list node structure */

typedef struct node
{
    student data;
    struct node *next;
} NODE;

/* function for inserting an element into a linked list */
void insertion(NODE ** start,student x)
{
    NODE * ptr, *prev,*p;
    ptr=*start;
    prev=NULL;
    p=(NODE *)malloc(sizeof(NODE));
    p->data=x;

    /* finding the position of x */
    while(ptr!=NULL)
        if(x.roll<ptr->data.roll)
            break;
        else
            {
                prev=ptr;
                ptr=ptr->next;
            }
}
```

```
if(prev==NULL)
    *start=p;
else
    prev->next=p;

    p->next=ptr;
}
void deletion(NODE ** start,student x)
{
    NODE * ptr, *prev,*p;

    /* searching for x */
    ptr=*start;
    prev=NULL;
    while(ptr!=NULL)
        if(x.roll==ptr->data.roll)
            break;
        else
            {
                prev=ptr;
                ptr=ptr->next;
            }
    if(ptr==NULL){printf("NOT present"); return;}
    if(prev==NULL)
        *start=ptr->next;
    else
        prev->next=ptr->next;

    free(ptr);
}

void display(NODE * start)
{
    if(start==NULL)printf("NULL");
    else
```

```
while(start!=NULL)
{
    printf("\n%s %d    ",start->data.name,start->data.roll);
    start=start->next;
}
printf("\n\n");
}

void edit(NODE * start,student x)
{ NODE * ptr;
ptr=start;
while(ptr!=NULL)
if(x.roll==ptr->data.roll)
    strcpy(ptr->data.name,x.name);return;
else ptr=ptr->next;

printf("Requested student not present");

}

void main()
{
NODE *start=NULL;
int n,i;
int c;
student x;
do
{ clrscr();
printf("Enter your choice\n");
printf("1      To insert\n");
printf("2      To delete\n");
printf("3      To edit\n");
printf("4      To display\n");
printf("5      To quit\n");
scanf("%d",&c);
```

(27) W

```
switch(c)
{
    case 1:printf("enter name and roll number of student\n");
    scanf("%s %d",x.name,&x.roll);
    insertion(&start,x);
    printf("\npress any key to continue");
    getch();
    break;

    case 2:printf("enter roll number of student\n");
    scanf("%d",&x.roll);
    deletion(&start,x);
    printf("\npress any key to continue");
    getch();
    break;

    case 3: printf("Enter roll no:of the student to be edited");
    scanf("%d",&x.roll);
    printf("Enter name");
    scanf("%s",x.name);
    edit(start,x);
    printf("\npress any key to continue");
    getch();
    break;
    case 4:clrscr();
    printf("          list of students\n\n");
    display(start);
    printf("\npress any key to continue");
    getch();
    break;
    default: exit(1);
}
while(c!=5);
```

- 27) WAP to find the largest and smallest elements of a given linked list.

(28)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
/*program to find largest and smallest elements in a linked list
*/
/*linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
}NODE;

/* type declaration of linked list */

typedef struct
{
    NODE * start;
}ll;

void add(ll * q,int x)
{
    NODE * ptr,*p;
    ptr=(NODE *)malloc(sizeof(NODE));
    ptr->data=x;ptr->next=NULL;
    if(q->start==NULL)q->start=ptr;
    else
        {p=q->start;
        while(p->next!=NULL)
            p=p->next;
        p->next=ptr;
        }
}

int maxl(ll q)
{
    NODE *p;
    int largest;
    if(q.start==NULL) return -1;

    largest=q.start->data;
    p=q.start->next;
```

```
        while(p!=NULL)
        {
            if(p->data>largest) largest=p->data;
            p=p->next;
        }
        return largest;
    }

int minl(ll q)
{
    NODE *p;
    int smallest;
    if(q.start==NULL) return -1;
    smallest=q.start->data;
    p=q.start->next;
    while(p!=NULL)
    {
        if(p->data<smallest) smallest=p->data;
        p=p->next;
    }
    return smallest;
}

void display(ll q)
{
    NODE *p;
    p=q.start;
    if(p==NULL) printf("LIST IS EMPTY");
    else
        while(p!=NULL)
        {
            printf("%d  ",p->data);
            p=p->next;
        }
}

void main()
{
    ll ql;
    int x,c;
    ql.start=NULL;
```

```
do
{ clrscr();
printf("Enter your choice\n");
printf("1      To add an element\n");
printf("2      To find max element\n");
printf("3      To find min element\n");
printf("4      To display\n");
printf("5      To quit\n");
scanf("%d",&c);
switch(c)
{
case 1:printf("Enter the element to be added");
scanf("%d",&x);
add(&q1,x);
printf("LIST AFTER ADDITION\n");
display(q1);
getch ();
break;
case 2:x=max1(q1);
if(x<0)printf("List is empty");
else
printf("MAX element %d\n",x);
getch ();
break;
case 3:x=min1(q1);
if(x<0)printf("List is empty");
else
printf("MIN element %d\n",x);
getch ();
break;
case 4: printf("LIST is: ");
display(q1);
getch ();
break;
default: exit(1);
}
}
while(c!=5);
```

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

/* to find ncr recursively */

int binom(int n,int r)
{
    if (r==0 || r==n) return 1;
    else
        return binom(n-1,r)+binom(n-1,r-1);
}

void main()
{
    int n,r;
    clrscr();
    printf("enter the values of n & r");
    scanf("%d%d",&n,&r);
    printf("binom=%d",binom(n,r));
    getch();
}
```

1) WAP to convert an infix expression into  
postfix form.

(32)

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
/* program to convert an infix expression into postfix expression
*/
/* stack structure */
typedef struct
{
    char a[100];
    int top;
}stack;

void push(stack * s, char x)
{
    s->a[++s->top]=x;
}

char pop(stack * s)
{
    char x;
    x=s->a[s->top--];
    return x;
}
int precede(char c1, char c2)
{
    switch (c1)
    {
        case '+': if (c2=='+' || c2=='') return 1;
                    else return 0;
        case '*': if (c2=='(') return 0;
                    return 1;
        case '(': return 0;
    }
}
void conversion(char in[], char post[])
{
    stack s1;
    int i, j;
    s1.top=-1;
    for(i=0, j=0; in[i]!='\0'; ++i)
    {
        if(isalpha(in[i])) post[j++]=in[i];
    }
}
```

```
        else
        {
            while(s1.top>=0 && precede(s1.a[s1.top],in[i]))
                post[j++]=pop(&s1);
            if(in[i]==')')s1.top--;
            else
                push(&s1,in[i]);
        }
    }
    while(s1.top>=0)
    {
        post[j++]=pop(&s1);
    }
    post[j]='\0';
}
void main()
{
    char postfix[80],infix[80];
    clrscr();
    printf("Enter a legal infix expression\n");
    gets(infix);
    conversion(infix,postfix);
    printf("postfix expression =");
    puts(postfix);
    getch ();
}
```

WAP to convert a postfix expression into infix form.

(34)

```
include <stdio.h>
include <conio.h>
include <string.h>
include <ctype.h>
* program to convert a postfix expression into equivalent infix
exp */

/* stack of strings */

typedef struct
{
    char s[50][100];
    int top;
}stack;

void push(stack *s1,char st[])
{
    strcpy(s1->s[++s1->top],st);
}

void pop(stack *s1,char st[])
{
    strcpy(st,s1->s[s1->top--]);
}

void conv(char post[],char in[])
{
    int i;
    char st[80],st1[80],st2[80],s[10];
    stack s1;

    for(i=0;post[i]!='\0';++i)
    { s[0]=post[i];s[1]='\0';
        if(isalpha(post[i]))push(&s1,s);
```

```
        else
        { pop(&s1,st1);
          pop(&s1,st2);
          strcpy(st,"(");
          strcat(st,st2);
          strcat(st,s);
          strcat(st,st1);
          strcat(st,")");
          push(&s1,st);
        }
      }
    pop(&s1,in);
}
void main()
{
  char post[80],in[80];
  clrscr();
  printf("ENTER A VALID POSTFIX STRING\n");
  gets(post);
  conv(post,in);
  printf("REQUIRED INFIX STRING: ");
  puts(in);
  getch();
}
```

13) WAP to maintain a circular queue using array.

(36)

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

/* program to maintain a circular queue using array */

/* type declaration of circular queue */

typedef struct
{
    int a[100];
    int front;
    int rear;
    int c;
}QUEUE;

void addq(QUEUE * q,int x)
{
    if(q->c==100)printf("QUEUE full");
    else
        { q->rear=(q->rear+1)%100;
          q->a[q->rear]=x;
          q->c++;
        }
}
int deleteq(QUEUE * q)
{
    int x;
    if(q->c==0)(printf("Q empty"));return -1;
    else
    {
        q->front=(q->front +1)%100;
        x=q->a[q->front];
        q->c--;
        return x;
    }
}
void display(QUEUE q)
{
    int i,j;
    if(q.c==0)printf("EMPTY QUEUE");
    else
        for(i=1,j=(q.front+1)%100;i<=q.c;i++,j=(j+1)%100)
            printf("%d ",q.a[j]);
```

```
void main()
{
    QUEUE q1;
    int x,c;
    q1.front=q1.rear=q1.c=0;
    do
    { clrscr();
        printf("Enter your choice\n");
        printf("1      To add\n");
        printf("2      To delete\n");
        printf("3      To display\n");
        printf("4      To quit\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:printf("Enter the element to be inserted");
            scanf("%d",&x);
            addq(&q1,x);
            printf("QUEUE AFTER ADDITION\n");
            display(q1);
            getch();
            break;

            case 2:x=deleteq(&q1);
            if(x>=0)printf("element deleted %d\n",x);
            getch();
            break;

            case 3: printf("Queue is: ");
            display(q1);
            getch();
            break;
            default: exit(1);
        }
    }
    while(c!=4);
}
```

WAP to maintain a circular queue using  
a linked list — Queue using circular linked  
list.

(38)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
/* program to maintain circular queue using linked list */

/*linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
}NODE;
/* circular queue structure */
typedef struct
{
    NODE *rear;
}cq;

void addq(cq * q,int x)
{
    NODE * ptr,*p,*prev;
    ptr=(NODE *)malloc(sizeof(NODE));
    ptr->data=x;
    if(q->rear==NULL)
        {q->rear=ptr;
         ptr->next=ptr;
    }
    else
    {
        ptr->next=q->rear->next;
        q->rear->next=ptr;
        q->rear=ptr;
    }
}

int deleteq(cq *q)
{
    NODE * ptr;
    int x;
    if(q->rear==NULL){printf("Empty cq");
                        return -1;
                     }
```

```

    else
    {
        ptr=q->rear->next;
        x=ptr->data;
        if(ptr==q->rear) q->rear=NULL;
        else
            q->rear->next=ptr->next;
        free(ptr);
        return x;
    }
}
void display(cq q)
{
    NODE *p,*r;
    r=p=q.rear->next;
    if(p==NULL)printf("CQ IS EMPTY");
    else
        do
        {
            printf("%d ",p->data);
            p=p->next;
        }
        while(p!=r);
}
void main()
{
    cq ql;
    int x,c;
    ql.rear=NULL;
    do
    {
        clrscr();
        printf("Enter your choice\n");
        printf("1      To add\n");
        printf("2      To delete\n");
        printf("3      To display\n");
        printf("4      To quit\n");
        scanf("%d",&c);

```

(40)

```
switch(c)
{
    case 1:printf("Enter the element to be inserted");
              scanf("%d",&x);
              addq(&ql,x);
              printf("CQ AFTER ADDITION\n");
              display(ql);
              getch();
              break;

    case 2:x=deleteq(&ql);
              if(x>=0)printf("element deleted %d\n",x);
              getch();
              break;

    case 3: printf("CQ is: ");
              display(ql);
              getch();
              break;
    default: exit(1);
}
while(c!=4);
```

15) WAP to maintain priority queue using linked list.

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

/* program to maintain priority queue using linked list */

/* linked list node structure */

typedef struct node
{
    int data;
    struct node *next;
} NODE;

/* type declaration of priority queue */

typedef struct
{
    NODE * front;
} pq;

void pqinsert(pq * q, int x)
{
    NODE * ptr, *p, *prev;
    ptr=(NODE *)malloc(sizeof(NODE));
    ptr->data=x;
    p=q->front; prev=NULL;
    while(p!=NULL)
        if(x<p->data)break;
        else
            (prev=p;
            p=p->next;
            )
    if(prev==NULL) {
        ptr->next=::front;
        q->front=ptr;
        }
    else
        {
            prev->next=ptr;
            ptr->next=p;
            }
}
```

```
A  
int pqdelete(pq *q)  
{  
    NODE * ptr;  
    int x;  
    if(q->front==NULL){printf("Empty pq");  
        return -1;  
    }  
  
    else  
    {  
        ptr=q->front;  
        x=ptr->data;  
        q->front=ptr->next;  
        free(ptr);  
        return x;  
    }  
}  
void display(pq q)  
{  
    NODE *p;  
    p=q.front;  
    if(p==NULL)printf("PQ IS EMPTY");  
    else  
        while(p!=NULL)  
        {  
            printf("%d ",p->data);  
            p=p->next;  
        }  
}  
void main()  
{  
    pq q1;  
    int x,c;  
    q1.front=NULL;  
    do  
    { clrscr();
```

```
printf("Enter your choice\n");
printf("1      To add\n");
printf("2      To delete\n");
printf("3      To display\n");
printf("4      To quit\n");
scanf("%d", &c);
switch(c)
{
    case 1:printf("Enter the element to be inserted");
              scanf("%d", &x);
              pqinsert(&q1, x);
              printf("PQ AFTER ADDITION\n");
              display(q1);
              getch();
              break;
    case 2:x=pqdelete(&q1);
              if(x>=0)printf("element deleted %d\n", x);
              getch();
              break;

    case 3: printf("PQ is: ");
              display(q1);
              getch();
              break;
    default: exit(1);
}
}
while(c!=4);
```

WAP to traverse a binary tree in inorder, preorder, and postorder. Find the no: of nodes, no: of leaf nodes and height of the tree.

(44)

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
typedef struct node
{
    int data;
    struct node *left,*right;
}treenode;

/* function to create a binary search tree */

treenode * createtree(int a[],int n)
{
    treenode * root,*ptr,*p,*prev;
    int i;
    root = NULL;
    for(i=0;i<n;++i)
    {
        ptr=(treenode *)malloc(sizeof(treenode));
        ptr->data=a[i];
        ptr->left=ptr->right=NULL;
        if(root==NULL)
            root=ptr;
        else
        {
            p=root;
            while(p!=NULL)
            {
                prev=p;
                if(a[i]<p->data)
                    p=p->left;
                else
                    p=p->right;
            }
            if(a[i]<prev->data)
                prev->left=ptr;
            else
                prev->right=ptr;
        }
    }
    return root;
}
```

```
void inorder(treenode * root)
{
    if(root!=NULL)
        {inorder(root->left);
         printf("%d ",root->data);
         inorder(root->right);
        }
}
void preorder(treenode * root)
{
    if(root !=NULL)
        {printf("%d ",root->data);
         preorder(root->left);
         preorder(root->right);
        }
}
void postorder(treenode * root)
{
    if(root !=NULL)
        {postorder(root->left);
         postorder(root->right);
         printf("%d ",root->data);
        }
}
/*function to find height of a binary tree */

int height(treenode * root)
{
    int l,r;
    if(root==NULL) return 0;
    l=l+height(root->left);
    r=r+height(root->right);
    if (l>r) return l;
    else return r;
}
```

```

/* function to count the number of leafnodes */
int leafcount(treenode * root)
{
    if (root == NULL) return 0;
    if (root->left==NULL && root->right==NULL) return 1;

    l=leafcount(root->left);
    r=leafcount(root->right);
    return l+r;
}

/*function to find number of nodes in a binary tree */

int nodecount(treenode * root)
{
    int l,r;
    if (root == NULL) return 0;
    return 1+nodecount(root->left)+nodecount(root->right);
}

void main()
{
    int a[100],n,i;
    treenode * root;
    clrscr();
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("enter elemenets\n");
    for(i=0;i<n;++i)
        scanf("%d",&a[i]);
    root=createtree(a,n); /*creating a binary search tree */
    clrscr();
    printf("\n\ninorder :data in ascending order\n");
    inorder(root);
    printf("\n\npreorder:\n");
    preorder(root);
    printf("\n\npostorder :\n");
    postorder(root);
    printf("\n\nHeight of the tree %d",height(root));
    printf("\n\nNumber of leaf nodes %d",leafcount(root));
    printf("\n\nNumber of nodes %d",nodecount(root));
    getch .;
}

```

(7) WAP to Create a copy and mirror image of a binary tree.

(47)

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

/* program to create a copy of a tree and its mirror image */

typedef struct node
{
    int data;
    struct node *left,*right;
}treenode;
treenode * createtree(int a[],int n)
{
    treenode * root,*ptr,*p,*prev;
    int i;
    root = NULL;
    for(i=0;i<n;++i)
    {
        ptr=(treenode *)malloc(sizeof(treenode));
        ptr->data=a[i];
        ptr->left=ptr->right=NULL;
        if(root==NULL)
            root=ptr;
        else
        {
            p=root;
            while(p!=NULL)
            {
                prev=p;
                if(a[i]<p->data)p=p->left;
                else p=p->right;
            }
            if(a[i]<prev->data)prev->left=ptr;
            else prev->right=ptr;
        }
    }
    return root;
}
```

```
void inorder(treenode * root)
{
    if(root !=NULL)
        {inorder(root->left);
        printf("%d    ",root->data);
        inorder(root->right);
        }
}
treenode * copy(treenode * root) /*function to create copy of a
tree*/
{
    treenode * p,*l,*r;
    p=NULL;
    if(root !=NULL)
        {
        l=copy(root->left);
        r=copy(root->right);
        p=(treenode *)malloc(sizeof(treenode));
        p->data=root->data;
        p->left=l;
        p->right=r;
        }
    return p;
}
treenode * mirror(treenode * root) /*function to create mirror
image*/
{
    treenode * p,*l,*r;
    p=NULL;
    if(root !=NULL)
        {
        l=mirror(root->left);
        r=mirror(root->right);
        p=(treenode *)malloc(sizeof(treenode));
        p->data=root->data;
        p->left=r;
        p->right=l;
        }
    return p;
}
```

```
void main()
{
    int a[100],n,i;
    treenode * root,*root1,*root2;
    clrscr();
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("enter elements\n");
    for(i=0;i<n;++i)
        scanf("%d",&a[i]);
    root=createtree(a,n);
    printf("inorder :data in the given tree\n");
    inorder(root);
    root1=copy(root);/*creating copy of the tree */
    printf("\ninorder :data in the copy of the tree\n");
    inorder(root1);
    root2=mirror(root);/*creating mirror image of the tree*/
    printf("\ninorder :data in the mirror image of the tree\n");
    inorder(root2);
    getch();
}
```

(f) WAP to insert and delete an element.

(50)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
/*program for inserting elements into a binary search tree and
deleting
elements from a binary search tree */
typedef struct node
{
    int data;
    struct node *left,*right;
}treenode;
void inorder(treenode *root)
{
    if(root!=NULL)
        (inorder(root->left);
        printf("%d\n",root->data);
        inorder(root->right);
    )
}
void insert(treenode ** root,int x)
{
    treenode *p,*ptr,*prev;
    ptr=(treenode *)malloc(sizeof(treenode));
    ptr->data=x;
    ptr->left=ptr->right=NULL;
    p=*root;
    prev=NULL;
    while(p!=NULL)
    {
        if(x<p->data) p=p->left;
        else
            p=p->right;
    }
    if(prev==NULL)*root=ptr;
    else
        if(x<prev->data) prev->left=ptr;
        else prev->right=ptr;
}
void deletion(treenode ** root,int x)
{
    treenode *p,*prev,*ptr,*rp,*s,*f;
    p=*root;
    prev=NULL;
```

```
while(p!=NULL)
    if(x==p->data)break;
    else{
        prev=p;
        if(x<p->data)p=p->left;
        else p=p->right;
    }
if(p==NULL)printf("Element is not present");
else
{
    if(p->left==NULL) rp=p->right;
    else
        if(p->right==NULL) rp=p->left;
        else
            { f=p; rp=p->right; s=rp->left;
                while(s!=NULL)
                    { f=rp;
                        rp=rp->left;
                        s=s->left;
                    }
                if(f!=p)
                    { f->left=rp->right;
                        rp->right=p->right;
                    }
                rp->left=p->left;
            }
    if(prev==NULL)
        *root=rp;
    else
        if(prev->right==p)prev->right=rp;
        else prev->left=rp;
    free(p);
}
```

```
void main()
{
treenode *root=NULL;
int c,x;
do
{
    clrscr();
    printf("                ENTER YOUR CHOICE\n");
    printf("                1: for INSERT\n");
    printf("                2: for DELETE\n");
    printf("                3: for DISPLAY\n");
    printf("                4: for EXIT\n");
    scanf("%d", &c);
    switch(c)
    {
        case 1:printf("enter element to be inserted");
                  scanf("%d",&x);
                  insert(&root,x);
                  printf("tree after insertion\n");
                  inorder(root);
                  break;

        case 2: if(root==NULL)printf("Empty TREE\n");
                  else{
                  printf("element to be deleted");
                  scanf("%d",&x);
                  deletion(&root,x);
                  printf("\ntree after deletion\n");
                  inorder(root);}
                  break;
        case 3: if(root==NULL)printf("Empty TREE\n");
                  else{
                  printf("tree elements in ascending order\n");
                  inorder(root); }
                  break;
        default:exit(1);
    }
    getch();
}
while(c!=4);
}
```

19) WAP to create an expression tree from a given postfix expression.

(53)

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
typedef struct node
{
    char data;
    struct node *left,*right;
}treenode;

/* function to create an expression tree from a given
postfix expression */

/*stack of pointers */
typedef struct
{
    treenode * a[100];
    int top;
}stack;

void push(stack * s,treenode *x)
{
    s->a[++s->top]=x;
}

treenode * pop(stack *s)
{
    treenode * x;
    x=s->a[s->top--];
    return x;
}

treenode * create(char post[])
{
    treenode *p,*p1,*p2;
    stack s1;
    int i;
    for(i=0;post[i]!='\0';++i)
    {
        p=(treenode *)malloc(sizeof(treenode));
        p->data=post[i];
        if(isalpha(post[i]))
            (p->left=p->right=NULL;
            push(&s1,p);
        }
    }
}
```

```
else
    {
        p2=pop(&s1);
        p1=pop(&s1);
        p->left=p1;
        p->right=p2;
        push(&s1,p);
    }
}
p=pop(&s1);
return p; /* returns the root of the created tree */
}
void inorder(treenode * root)
{
    if(root !=NULL)
        {if(root->left!=NULL && root->right!=NULL)
            printf("(");
            inorder(root->left);
            printf("%c",root->data);
            inorder(root->right);
            if(root->left!=NULL && root->right!=NULL)
                printf(")");
        }
}
void preorder(treenode * root)
{
    if(root !=NULL)
        {printf("%c" ,root->data);
        preorder(root->left);
        preorder(root->right);
        }
}
void postorder(treenode * root)
{
    if(root !=NULL)
        {postorder(root->left);
        postorder(root->right);
        printf("%c",root->data);
        }
}
```

```
void main()
{
    char postfix[100];
    treenode * root;
    clrscr();
    printf("Enter a postfix string\n");
    gets(postfix);
    root=create(postfix);
    printf("\n\ninfix expression :");
    inorder(root);
    printf("\n\nprefix expression:");
    preorder(root);
    printf("\n\npostfix expression:");
    postorder(root);
    getch();
}
```

20) WAP to evaluate an expression tree.

(56)

```
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
/* program to evaluate a postfix expression using expression tree */
typedef struct node
{
    char data;
    struct node *left,*right;
}treenode;

/* function to create an expression tree from a given
postfix expression */
typedef struct
{
    treenode * a[100];
    int top;
}stack;
void push(stack * s,treenode *x)
{
    s->a[++s->top]=x;
}
treenode * pop(stack *s)
{
    treenode * x;
    x=s->a[s->top--];
    return x;
}
treenode * create(char post[])
{
    treenode *p,*p1,*p2;
    stack s1;
    int i;
    for(i=0;post[i]!='\0';++i)
    {
        p=(treenode *)malloc(sizeof(treenode));
        p->data=post[i];
        if(isdigit(post[i]))
            (p->left=p->right=NULL);
        push(&s1,p);
    }
}
```

```
else
{
    p2=pop(&s1);
    p1=pop(&s1);
    p->left=p1;
    p->right=p2;
    push(&s1,p);
}
}
p=pop(&s1);
return p; /* returns the root of the created tree */
}
int operation( char op,int p1,int p2)
{
    switch(op)
    {
        case '+' :return p1+p2;
        case '*' :return p1*p2;
        case '-' :return p1-p2;
    }
}
int eval(treenode * root)
{ int l,r;
    if (root==NULL) return 0;
    if(root->left==NULL && root->right==NULL)
        return root->data-'0';
    l=eval(root->left);
    r=eval(root->right);
    return operation(root->data,l,r);
}
void main()
{
    char postfix[100];
    treenode * root;
    clrscr();
    printf("Enter a postfix string operands should be digits\n");
    printf("e.g., 12+ means 1+2\n");
    gets(postfix);
    root=create(postfix);
    printf("value of the given expression= %d",eval(root));
    getch();
}
```

Q1) WAP to sort an array using bubble sort.

(58)

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

/* program to implement bubble sort */

void bubble(int a[], int n)
{
    int i, pass=1, exchange=1, temp;
    while(pass<=n-1 && exchange)
    {
        exchange=0;
        for(i=0; i<n-pass; ++i)
            if(a[i]>a[i+1])
                {temp=a[i];
                 a[i]=a[i+1];
                 a[i+1]=temp;
                 exchange=1;
                }
        printf("Array after %d pass ", pass);
        for(i=0; i<n; ++i)
            printf("%5d", a[i]);
        printf("\n\n");
        pass++;
    }
    printf("Total number of passes %d\n\n", pass-1);
}

void main()
{
    int a[100], n, i;
    clrscr();
    printf("                                BUBBLE SORT \n\n");
    printf("Enter Number of Elements ");
    scanf("%d", &n);
    printf("Enter Elements \n");
    for(i=0; i<n; ++i)
        scanf("%d", &a[i]);

    bubble(a, n);
    printf("\n\n      sorted list\n\n");
    for(i=0; i<n; ++i)
        printf("%5d", a[i]);
    getch();
}
```

22) WAP to sort an array using insertion sort  
and shell sort.

(59)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
/* program to sort an array in ascending order using insertion
sort and
Shell sort */

void insort(int a[], int n)
{
    int i, k, x;
    for(k=1; k<n; ++k)
    { x=a[k];
        for(i=k-1; i>=0 && x<a[i]; -- i)
        {
            a[i+1]=a[i];
        }
        a[i+1]=x; /* inserting x */
    }
}
void shellsort(int a[], int n)
{
    int i, k, x, j, width;

    width=n/2;
    while(width>=1)
    {
        for(k=width; k<n; k++)
        { x=a[k];
            for(i=k-width; i>=0 && x<a[i]; i-=width)
            {
                a[i+width]=a[i];
            }
            a[i+width]=x;

        }
        width=width/2;
    }
}
void main()
{int a[100], n, i, x;
 clrscr();
```

```
printf("INSERTION SORT\n");
printf("enter number of elements\n");
scanf("%d", &n);
printf("enter elements\n");
for(i=0;i<n;++)
    scanf("%d", &a[i]);

insort(a,n);

printf("sorted list\n");
for(i=0;i<n;++)
    printf("%d ", a[i]);
printf("\n\nSHELL SORT\n");
printf("enter number of elements\n");

scanf("%d", &n);
printf("enter elements\n");
for(i=
0;i<n;++)
    scanf("%d", &a[i]);
shellsort(a,n);
printf("sorted list\n");
for(i=0;i<n;++)
    printf("%d ", a[i]);

getch();
}
```

23) WAP to sort an array using selection sort

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

/* program to implement selection sort */

void selsort(int a[], int n)
{
    int i, j, pos, max;
    for(j=n-1; j>=1; --j)
    {
        /* finding largest of a[0..j] */
        max=a[0];
        pos=0;
        for(i=1; i<=j; ++i)
            if(a[i]>max)
                {max=a[i];
                 pos=i;
                }
        /* exchanging largest & last elements of a[0..j] */
        /* last is a[j] and largest is a[pos] */

        a[pos]=a[j];
        a[j]=max;
    }
}

void main()
{
    int a[100], n, i;
    clrscr();
    printf("                      SELECTION SORT \n\n");
    printf("Enter Number of Elements ");
    scanf("%d", &n);
    printf("Enter Elements \n");
    for(i=0; i<n; ++i)
        scanf("%d", &a[i]);

    selsort(a, n);
    printf("\n\n      sorted list\n\n");
    for(i=0; i<n; ++i)
        printf("%5d", a[i]);
    getch();
}
```

24) WAP to sort an array using radix sort.  
(Linked List version)

52

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

/* program to sort an array using radix sort LINKED LIST
implementation */

void radix(int a[], int n, int m)
{
    typedef struct node
    {
        int data;
        struct node * next;
    }NODE;

    NODE * ptr, *start, *prev;
    NODE *front[10], *rear[10];
    int k=1, i, j, y, p;;
    /*creating initial linked list*/
    start=NULL;
    for(i=0; i<n; ++i)
    {
        ptr=(NODE *)malloc(sizeof(NODE));
        ptr->data=a[i];
        ptr->next=NULL;
        if(start==NULL)
            start=ptr;
        else
            prev->next=ptr;
        prev=ptr;
    }

    /*radix sort*/
    for(i=1; i<=m; ++i)
    {
        for(j=0; j<10; ++j)
            front[j]=NULL;
        /*placing elements into queues*/
        ptr=start;
        while(ptr!=NULL)
        {
            y=ptr->data/k % 10; /*y is the digit*/
            if(front[y]==NULL)
                front[y]=ptr;
            rear[y]=ptr;
        }
    }
}
```

```
        else {
            rear[y]->next=ptr;
            rear[y]=ptr;
        }

        ptr=ptr->next;
    }
/*creating new linked list*/
start=NULL;
for(j=0;j<10;++j)
    if(front[j]!=NULL){if(start==NULL)start=front[j];
                        else rear[p]->next=front[j];
                        p=j;
    }
    rear[p]->next=NULL;
    k=k*10;
}
/*copying back to array*/
ptr=start;
for(i=0;i<n;++i,ptr=ptr->next)
    a[i]=ptr->data;

}

void main()
{
    int a[100],n,i,m;
    clrscr();
    printf(" RADIX SORT\n");
    printf("Enter number of numbers and number of digits\n");
    scanf("%d%d",&n,&m);
    printf("enter elements\n");
    for(i=0;i<n;++i)
        scanf("%d",&a[i]);
    radix(a,n,m);
    printf("sorted list\n");
    for(i=0;i<n;++i)
        printf("%d ",a[i]);
    getch ();
}
```

(63) 5) WAP to sort an array using radix sort  
( Array version)

(64)

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

/* program to sort an array using radix sort ARRAY
implementation */

void radix(int a[], int n, int m)
{
    int q[10][30], rear[10];
    int i, j, k=1, s, l, digit;

    for(i=1; i<=m; ++i)
    {
        /*initializing queues */
        for(j=0; j<10; ++j)
            rear[j]=-1;

        /*placing elements of a in queues */

        for(j=0; j<n; ++j)
        {
            digit=a[j]/k%10;
            q[digit][++rear[digit]]=a[j];
        }
        /*copying elements of q[0],q[1],q[2]...q[9] into array a */
        s=0;
        for(j=0; j<10; ++j)
            for(l=0; l<=rear[j]; ++l)
                a[s++]=q[j][l];

        k=k*10;
    }
}
```

```
void main()
{
    int a[100],n,i,m;
    clrscr();
    printf(" RADIX SORT\n");
    printf("Enter number of numbers and number of digits\n");
    scanf("%d%d",&n,&m);
    printf("enter elements\n");
    for(i=0;i<n;++i)
        scanf("%d",&a[i]);
    radix(a,n,m);
    printf("sorted list\n");
    for(i=0;i<n;++i)
        printf("%d ",a[i]);
    getch ();
}
```

WAP to sort an array using bucket sort.

(66)

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

/* program to implement bucket sort */

void bucketsort(int a[], int n, int m)
{
    int *count, i, j, k=0;
    count=(int *)calloc(sizeof(int),m);
    for(i=0; i<n; ++i)
        count[a[i]]++;
    for(i=0; i<m; ++i)
        for(j=1; j<=count[i]; ++j)
            a[k++]=i;
}

void main()
{
    int a[100], n, i, max;
    clrscr();
    printf("BUCKET SORT\n");
    printf("enter number of elements\n");
    scanf("%d", &n);
    printf("enter the elements\n");
    for(i=0; i<n; ++i)
    {
        scanf("%d", &a[i]);
    }
    /*finding largest element of a */
    max=a[0];
    for(i=1; i<n; ++i)
    {
        if(a[i]>max) max=a[i];
    }
    bucketsort(a, n, max+1);
    printf("sorted list is\n");
    for(i=0; i<n; ++i)
        printf("%d ", a[i]);
    getch();
}
```

27) WAP to sort an array using heap sort.

(67)

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

/* program to sort an array using heap sort */
void swap(int *x, int *y)
{
    int temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

void heapsort(int a[], int n)
{
    int i, s, f;

    /* creating initial heap */

    for (i = 1; i < n; ++i)
    {
        s = i;
        f = (s - 1) / 2;
        while (a[f] < a[s])
            { swap(&a[f], &a[s]); }
        s = f;
        if (s == 0) break;
        f = (s - 1) / 2;
    }

    /* sorting */
    for (i = n - 1; i >= 1; --i)
    {
        swap(&a[0], &a[i]);
        f = 0;
        s = 1;
        if (i == 1) break;
        if (i > 2) if (a[2] > a[1]) s = 2;
    }
}
```

```
while( a[f]<a[s] )
{
    swap(&a[f],&a[s]);
    f=s;
    s=2*f+1;
    if(i>s+1) if(a[s+1]>a[s]) s=s+1;
    if (s>=i) break;
}
}

oid main()
{
    int a[100],n,i;
    clrscr();
    printf("          HEAP SORT\n");
    printf("\n\nenter the number of elements\n");

    scanf("%d",&n);

    printf("Enter elements\n");

    for(i=0;i<n;++i)
        scanf("%d",&a[i]);

    heapsort(a,n);

    printf("\n\nSorted List\n");

    for(i=0;i<n;++i)
        printf("%d    ",a[i]);

    getch ();
}
```

28) wAP to sort an array using mergesort.  
(Recursive implementation)

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

/* program to implement recursive merge sort */

void merge(int a[], int low, int mid, int high)
{
    int i, j, k, h;
    int b[100];
    h=low; i=low; j=mid+1;

    while(h<=mid && j<=high)
    {
        if(a[h]<a[j])
            b[i++]=a[h++];
        else
            b[i++]=a[j++];
    }
    while(h<=mid)

    {
        b[i++]=a[h++];
    }
    while(j<=high)
    {
        b[i++]=a[j++];
    }
    for(k=low; k<=high; ++k)
    {
        a[k]=b[k];
    }
}

void mergesort(int a[], int low, int high)
{
    int mid;
    if(low<high)
    {
        mid=(low+high)/2;
        mergesort(a, low, mid);
        mergesort(a, mid+1, high);
        merge(a, low, mid, high);
    }
}
```

```
oid main()

int a[100],n,i;
clrscr();
printf("          .. RECURSIVE MERGE SORT \n\n");
printf("Enter Number o f Elements ");
scanf("%d",&n);
printf("Enter Elements \n");
for(i=0;i<n;++i)
    scanf("%d",&a[i]);

mergesort(a,0,n-1);
printf("\n\n      sorted list\n\n");
for(i=0;i<n;++i)
    printf("%5d",a[i]);
getch();
}
```

2g) WAP to sort an array using mergesort.  
(Non-recursive implementation)

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

/* program to sort an array using merge sort - NON RECURSIVE
VERSION */

void mergesort(int a[], int n)
{
    int c[100], lb1, ub1, lb2, ub2, i, j, k, size=1;

    while(size<n)
    {
        lb1=0; k=0;
        while(lb1+size<n)
        {
            lb2=lb1+size;
            ub1=lb2-1;
            if(lb2+size-1<n)
                ub2=lb2+size-1;
            else
                ub2=n-1;
            /*merging a[lb1-ub1] and a[lb2-ub2]*/
            i=lb1; j=lb2;
            while(i<=ub1 && j<=ub2)
                if (a[i]<a[j]) c[k++]=a[i++];
                else c[k++]=a[j++];

            while(i<=ub1)
                c[k++]=a[i++];

            while(j<=ub2)
                c[k++]=a[j++];
            lb1=ub2+1;
        }/*end of while(lb1+size<n) */
        for(i=0;i<lb1; ++i)
            a[i]=c[i];
        size=2*size;
    }/*end of while (size<n) */
}/* end of the function */
```

71

```
void main()
{
    int a[100], n, i;
    clrscr();
    printf("MERGE SORT\n");
    printf("Enter number of elements\n");
    scanf("%d", &n);
    printf("Enter elements\n");
    for(i=0; i<n; ++i)
        scanf("%d", &a[i]);
    mergesort(a, n);
    printf("sorted list\n");
    for(i=0; i<n; ++i)
        printf("%d    ", a[i]);
    getch();
}
```

72

30 WAP to sort an array using quicksort.  
(Recursive and non-recursive)

(73)

```
#include <stdio.h>
#include <conio.h>
/* program to sort an array using quicksort - recursive & non
recursive */

int partition(int a[], int lb, int ub)
{
    int down, up, x, temp;
    down=lb; up=ub;
    x=a[lb];
    while(down < up)
    {
        while(a[down]<=x && down < ub)
            down++;
        while(a[up]>x) up--;
        if(down<up)
            {
                temp=a[down];
                a[down]=a[up];
                a[up]=temp;
            }
    }
    a[lb]=a[up];
    a[up]=x;
    return up;
}
void quicksort1(int a[], int lb, int ub)
{
    struct xyz
    {
        int l, u;
    };
    int top=0, j;
    struct xyz stack[100];

    stack[top].l=lb;
    stack[top].u=ub;
    while(top>=0)
    {
        lb=stack[top].l;
        ub=stack[top].u;
        top--;
    }
}
```

```
        while(lb<ub)
            {
                j=partition(a,lb,ub);
                top++;
                stack[top].l=j+1;
                stack[top].u=ub;
                ub=j-1;
            }
        }
void quicksort2(int a[],int lb,int ub)
{
    int j;
    if(lb<ub){
        j=partition(a,lb,ub);
        quicksort2(a,lb,j-1);
        quicksort2(a,j+1,ub);
    }
}
void main()
{
    int a[100],n,i,choice;
    clrscr();
    printf("          QUICK SORT\n");
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter elements\n");
    for(i=0;i<n;++i)
        scanf("%d",&a[i]);
    printf("\n\nRecursive OR Non-recursive?\n");
    printf("\nEnter 1 for Recursive 2 for Non-Recursive\n");
    scanf("%d",&choice);
    if(choice==1)
        quicksort2(a,0,n-1);
    else
        quicksort1(a,0,n-1);
    printf("Sorted list\n");
    for(i=0;i<n;++i)
        printf("%d ",a[i]);
    getch();
}
```

31) WAP to search for an element using sequential search.

```
(75)  
#include <stdio.h>  
#include <conio.h>  
/* program to implement sequential search */  
  
int seqsearch(int a[], int n, int x)  
{  
    int i;  
    for(i=0; i<n; ++i)  
        if(a[i]==x) return i;  
  
    return -1;  
}  
  
void main()  
{  
    int a[100], n, x, i, k;  
    clrscr();  
    printf("enter number of elements\n");  
    scanf("%d", &n);  
    printf("enter elements\n");  
    for(i=0; i<n; ++i)  
        scanf("%d", &a[i]);  
    do {  
        clrscr();  
        printf("given list of items\n");  
        for(i=0; i<n; ++i)  
            printf("%4d", i); printf("\n");  
        for(i=0; i<n; ++i)  
            printf("----");  
        printf("\n");  
        for(i=0; i<n; ++i)  
            printf("%4d", a[i]);  
        printf("\n\n")  
        printf("element to be searched to stop enter -1\n");  
        scanf("%d", &x);  
        if(x>=0) {  
            k=seqsearch(a, n, x);  
            if(k>=0) printf("present at a[%d]", k);  
            else printf("not present");  
        }  
        getch();  
    } while(x>=0);  
}
```

WAP to search for an element using binary search  
search( recursive implementation)

```
(38) (39)
#include <stdio.h>
#include <conio.h>

/* program to implement recursive binary search */
int binsearch(int a[], int low, int high, int x)
{
    int mid;
    if (low <= high) {
        mid = (low + high) / 2;
        if (x == a[mid]) return mid;
        if (x < a[mid]) return binsearch(a, low, mid - 1, x);
        else return binsearch(a, mid + 1, high, x);
    }
    else return -1;
}

void main()
{
    int a[100], n, x, i, k, flag;
    clrscr();
    printf("enter number of elements\n");
    scanf("%d", &n);
    do {flag = 0;
        printf("enter elements in ascending order\n");
        for (i = 0; i < n; ++i)
            scanf("%d", &a[i]);
        for (i = 0; i < n - 1; ++i)
            if (a[i] > a[i + 1]) (printf("elements are not in ascending
order\n"));
            flag = 1; getch();
            break;
    }
    clrscr();
} while (flag == 1);
do { clrscr();
printf("RECURSIVE BINARY SEARCH\n");
printf("given list of items\n");
for (i = 0; i < n; ++i)
    printf("%4d", a[i]); printf("\n");
}
```

```
for(i=0;i<n;++i)
    printf("----");
printf("\n");
for(i=0;i<n;++i)
    printf("%4d",a[i]);
printf("\n\n")element to be searched to stop enter -1\n");
scanf("%d",&x);
if(x>=0){k=binsearch(a,0,n-1,x);
    if(k>=0)printf("present at [a[%d]",k);
    else printf("not present");
}
getch ();
}while(x>=0);
```

(77)

WAP to search for an element using binary search  
(Non- recursive implementation)

(78)

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

/* NON RECURSIVE implementation of binary search */

int binsearch(int a[], int n, int x)

    int low, high, mid;
    low=0; high=n-1;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(x==a[mid]) return mid;
        if(x<a[mid]) high=mid-1;
        else low=mid+1;
    }
    return -1;

void main()

    int a[100], n, x, i, k, flag;
    clrscr();
    printf("enter number of elements\n");
    scanf("%d", &n);
    do(flag=0;
    printf("enter elements in ascending order\n");
    for(i=0; i<n; ++i)
        scanf("%d", &a[i]);
    for(i=0; i<n-1; ++i)
        if(a[i]>a[i+1]) {printf("elements are not in ascending
        order\n");
            flag=1; getch();
            break;
        }
    clrscr();
    while(flag==1);
    do{ clrscr();
    printf("BINARY SEARCH\n");
    printf("given list of items\n");
    }
```

```
for(i=0;i<n;++i)
    printf("%4d",i);printf("\n");
for(i=0;i<n;++i)
    printf("----");
printf("\n");
for(i=0;i<n;++i)
    printf("%4d",a[i]);
printf("\n\n")element to be searched to stop enter -1\n");
scanf("%d",&x);
if(x>=0)(k=binsearch(a,n,x));
    if(k>=0)printf("present at a[%d]",k);
        else printf("not present");
    }
getch();
}while(x>=0);
}
```

(79)   
 C program to search for an element using indexed sequential search.

(80)

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

program to implement INDEXED SEQUENTIAL SEARCH */
kindex[10], pindex[10], indexsize;
indseqsearch(int a[], int n, int x)

int i, low, high;
for(i=0; i<indexsize; ++i)
    if(x<kindex[i]) break;
if (i==0) low=0; else low=pindex[i-1];
if (i==indexsize) high=n-1; else high=pindex[i]-1;
for(i=low; i<=high; ++i)
    if(x==a[i]) return i;

return -1;

d main()

int a[100], n, x, i, j, k, flag;
clrscr();
printf("enter number of elements >= 10\n");
scanf("%d", &n);
if(flag=0);
printf("enter elements in ascending order\n");
for(i=0; i<n; ++i)
    scanf("%d", &a[i]);
for(i=0; i<n-1; ++i)
    if(a[i]>a[i+1]) {printf("elements are not in ascending
order\n");
flag=1; getch();
break;
}
clrscr();
while(flag==1);
for(i=0, j=2; j<n; ++i, j+=3)
    (kindex[i]=a[j]);
    pindex[i]=j;
}
indexsize=i;
```

```
do{ clrscr();
printf("INDEXED SEQUENTIAL SEARCH\n");
printf("      given list of items\n");
for(i=0;i<n;++i)
    printf("%5d",i);printf("\n");
for(i=0;i<n;++i)
    printf("----");
printf("\n");
for(i=0;i<n;++i)
    printf("%5d",a[i]);
printf("\n\n      INDEX TABLE\n");
for(i=0;i<indexsize;++i)
    printf("%5d",i);printf("\n");
for(i=0;i<indexsize;++i)
    printf("----");
printf("\n");
for(i=0;i<indexsize;++i)
    printf("%5d",kindex[i]);

printf("\n\nelement to be searched to stop enter -1\n");
scanf("%d",&x);
if(x>=0) {k=indseqsearch(a,n,x);
    if(k>=0)printf("present at a[%d]",k);
    else printf("not present");
}
getch();
}while(x>=0);
}
```

35) WAP to implement interpolation search.

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

/*program to implement INTERPOLATION SEARCH */

int interpolation(int a[], int n, int x)
{
    int low, high, mid;
    low=0; high=n-1;
    while(x>=a[low] && x<=a[high])
    {
        if(low==high)
            mid=low;
        else
            mid=low+(high-low)*(x-a[low])/(a[high]-a[low]);
        if(x==a[mid]) return mid;
        if(x<a[mid]) high=mid-1;
        else low=mid+1;
    }
    return -1;
}

void main()
{
    int a[100], n, x, i, k, flag;
    clrscr();
    printf("enter number of elements\n");
    scanf("%d", &n);
    do{flag=0;
    printf("enter elements in ascending order\n");
    for(i=0; i<n; ++i)
        scanf("%d", &a[i]);
    for(i=0; i<n-1; ++i)
        if(a[i]>a[i+1]) (printf("elements are not in ascending
order\n"));
        flag=1; getch();
        break;
    }
    clrscr();
}while(flag==1);
```

(82)

```
do{ clrscr();
printf("INTERPOLATION SEARCH\n");
printf("given list of items\n");
for(i=0;i<n;++i)
    printf("%4d",i);printf("\n");
for(i=0;i<n;++i)
    printf("----");
printf("\n");
for(i=0;i<n;++i)
    printf("%4d",a[i]);
printf("\n\nelement to be searched to stop enter -1\n");
scanf("%d",&x);
if(x>=0){k=interpolation(a,n,x);
    if(k>=0)printf("present at a[%d]",k);
    else printf("not present");
}
getch ();
}while(x>=0);
}
```

WAP to implement hash search( hashing) using linear probing to resolve collision.

(84)

```
#include <stdio.h>
#include <conio.h>
/* program to implement hash search */

int hashsearch(int a[], int x, int n)
{
    int index, start;
    index = x % n;
    if (a[index] == x) return index;
    else if (a[index] == -1) return -1;
    else
        {start = index;
         do
            { index = (index + 1) % n;
              if (a[index] == x) return index;
              else if (a[index] == -1) break;
            }
         while (index != start);
        return -1;
       }
}
void main()
{int hsh[10], i, x, index, k;
 clrscr();
 printf("HASHING USING LINEAR PROBING\n\n");

 printf("      HASH table creation\n\n");
 for (i=0; i<10; ++i)
     hsh[i] = -1;
 for (i=1; i<=7; ++i)
 { printf("Enter a number\n");
   scanf("%d", &x);
   index = x % 10;
   while (hsh[index] != -1)
       index = (index + 1) % 10;
   hsh[index] = x;

  }
 clrscr();
 printf("      Hash Table\n");
}
```

```
for(i=0;i<10;++i)
    printf("%4d ",i);printf("\n");
for(i=0;i<10;++i)
    printf("%4d ",hsh[i]);
do{   printf("\nelement to be searched    to stop enter -1\n");
    scanf("%d",&x);
    if(x>=0){k=hashsearch(hsh,x,10);
        if(k>=0)

                printf("%d  is  present at hsh[%d]\n
",x,k);
            else printf("%d is NOT present\n" ,x);
        }
    }
while(x>=0);

getch ();
}
```

37) wAP to implement hashing using separate chaining.

(86)

92

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

/*Hash search linked list implementation (CHAINING METHOD) */

typedef struct node
{
    int data;
    struct node *next;
} NODE;
int flag;
NODE * hashsearch( NODE * a[],int x,int n)
{
    NODE * ptr,*prev,*p;
    int index;
    flag=0;
    index=x % n;
    ptr=a[index];prev=NULL;
    while(ptr!=NULL)
    {
        if(x==ptr->data){flag =1;return ptr;}
        else
            {prev=ptr;
             ptr=ptr->next;
            }
    }
    p=(NODE *)malloc(sizeof(NODE));
    p->data=x;
    p->next=NULL;
    if(prev==NULL)a[index]=p;
    else prev->next=p;
    return p;
}
void main()
{
    NODE * hsh[10],*ptr,*k;
    int i,x;
    for(i=0;i<10;++i)
        hsh[i]=NULL;
    clrscr();
```

```
do{ if(x>=0)
{
    clrscr();
    printf("          HASH TABLE \n\n");
    for(i=0;i<10;++i)
        (printf("\n");
         if(hsh[i]==NULL)printf("hsh[%d]--->\n",i);
         else {printf("hsh[%d]--->",i);
                ptr=hsh[i];
                while(ptr!=NULL)
                    { printf("%d-->",ptr->data);
                      ptr=ptr->next;
                    }
                printf("NULL\n");
            })
    }
    printf("Enter a number to stop enter -1\n");
    scanf("%d",&x);

    hashsearch(hsh,x,10);
}
}
while(x>=0);
printf("Element to be searched\n");
scanf("%d",&x);
k=hashsearch(hsh,x,10);
if(flag==1)printf("%d originally present at %u\n",x,k);
else printf("%d is added into the chain at %u\n",x,k);
getch ();
}
```

38) WAP to implement hashing using double hashing to resolve collision.

(88)

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

int hashsearch(int a[], int x, int n)
{
    int index, start;
    index=x%n;
    if(a[index]==x) return index;
    else if(a[index]==-1) return -1;
    else
        (start=index;
         do
            { index=(index+(1+x%(n-1)))%n;
              if(a[index]==x) return index;
              else if(a[index]==-1) break;
            }
         while(index != start));
    return -1;
}
void main()
{int hsh[11], i, x, index, k;
 clrscr();
 printf("HASHING USING DOUBLE HASHING\n\n");
 printf("      HASH table creation\n\n");
 for(i=0;i<11;++i)
     hsh[i]=-1;
 for(i=1;i<=7; ++i)
 { printf("Enter a number\n");
   scanf("%d", &x);
   index=x%11;
   while(hsh[index]!=-1)
       index=(index+(1+x%10))%11;
   hsh[index]=x;
 }
 do(clrscr();
 printf("      Hash Table\n");
 for(i=0;i<10; ++i)
   printf("%4d ", i);printf("\n");
 }
```

```
for(i=0;i<10;++i)
    printf("%4d ",hsh[i]);
    printf("\nelement to be searched to stop enter -1\n");
    scanf("%d",&x);
    if(x>=0){k=hashsearch(hsh,x,11);
        if(k>=0)
            printf("%d is present at hsh[%d]\n"
",x,k);
        else printf("%d is NOT present\n",x);
    }
    getch();
}
while(x>=0);

getch();
}
```

39) WAP to implement breadth first search (BFS)

(90)

ch (d)

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

#define INF 32767

/* program to implement bfs traversal */

int adj[10][10], n;
int visited[10];
void bfs(int v)
{
    int q[10], front=-1, rear=-1, i;
    visited[v]=1;
    q[++rear]=v;
    printf("visiting order \n");
    while(front!=rear)
    {
        v=q[++front];
        printf("%d ", v);
        for(i=0; i<n; ++i)
            if(!visited[i] && adj[v][i])
                { visited[i]=1;
                  q[++rear]=i;
                }
    }
}

void main()
{
    int i, j, m, a, b, v;
    char c;
    clrscr();
    printf("Enter number of nodes and number of edges\n");
    scanf("%d%d", &n, &m);
    for(i=0; i<n; ++i)
        for(j=0; j<n; ++j)
            adj[i][j]=0;
    for(i=1; i<=m; ++i)
        { printf("Enter an edge \n");
          scanf("%d%d", &a, &b);
          adj[a][b]=1; /*undirected graph */
          adj[b][a]=1;
        }
}
```

(q1)

```
do{ clrscr();
    printf("          Adjacency Matrix \n");
    for(i=0;i<n;++i)
    { for(j=0;j<n;++j)
        printf("%4d",adj[i][j]);
        printf("\n");
    }
    printf("Enter initial node\n");
    scanf("%d",&v);
    for(i=0;i<n;++i)visited[i]=0;
    bfs(v);
    fflush(stdin);
    printf("\nDo you want to continue? Enter Y or N\n");
    scanf("%c",&c);
}
while(c!='N');
getch();
}
```

40) WAP to implement depth first search (dfs).

(92)

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

/* dfs traversal */

void graph(int adj[][10], int n, int m)
{
    int i, j, a, b;
    for(i=0; i<n; ++i)
        for(j=0; j<n; ++j)
            adj[i][j]=0;
    for(i=0; i<m; ++i)
    {
        printf("enter an edge\n");
        scanf("%d%d", &a, &b);
        adj[a][b]=1;
        adj[b][a]=1;
    }
}
int visited[10]={0};
void dfs(int adj[][10], int n, int v)
{
    int i;
    visited[v]=1;
    printf("%d ", v);
    for(i=0; i<n; ++i)
        if(adj[v][i] && !visited[i])dfs(adj, n, i);
}
void main()
{
    int adj[10][10];
    int n, m, i, j, v;
    clrscr();
    printf("enter no: of nodes & no:of edges\n");
    scanf("%d%d", &n, &m);

    graph(adj, n, m);
    printf("Enter the initial node\n");
    scanf("%d", &v);
    dfs(adj, n, v);
    getch();
}
```

41) WAP to find the shortest path between two nodes of a network using Dijkstra's algorithm.

(93)

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

#define INF 32767

/* program to find shortest path between two nodes of a given
network using DIJKSTRA'S algorithm */

int adj[10][10], weight[10][10], n;
int shortpath(int s, int t, int path[])
{
    int i, dc, newdist, mindist, current;
    int dist[10], nodeset[10];
    for(i=0; i<n; ++i)
        (dist[i]=INF;
         nodeset[i]=0;
        )
    current=s;
    dist[current]=0;
    nodeset[current]=1;    /* selected */
    while(current != t)
    {
        dc=dist[current];
        for(i=0; i<n; ++i)
            if(adj[current][i] && !nodeset[i])
                { newdist=dc+weight[current][i];
                  if(newdist<dist[i])
                      {dist[i]=newdist;
                       path[i]=current;
                      }
                }
        /* finding node with minimum distance */
        mindist=INF;
        for(i=0; i<n; ++i)
            if(!nodeset[i] && dist[i]<mindist)
            {
                mindist=dist[i];
                current=i;
            }
        nodeset[current]=1;
    }
    return dist[t];
}
```

~~current~~