KIIT UNIVERSITY

2020

DSA Lab Record:2020

NISCHIT PUDSASAINI TO-DSA DEPARTMENT

ROLL NO-1905995

QUESTION NUMBER-1(ARRAYS)

- **Program to find repeating element in an array (duplicate elements)**

```c
//Searching of duplicate inside an array
#include<stdio.h>
#define MAX 6
int duplicate(int yt[]);
int main()
{
    int arr[MAX]={89,78,67,89,78,90};
    int f=duplicate(arr);
    if(0)
    {
        printf("\nduplicate not found in an array\n");
    }
    else
    {
        printf("\nFinding dulicate complete\n");
    }

    return 0;
}
int duplicate(int yt[])
{
    int x=0;
    int a=0;
    for(int i=0;i<MAX-1;i++)
    {
        a=a+1;
        for(int j=a;j<MAX-1;j++)
        {
            if(yt[i]==yt[j])
            {
                printf("\n%d\n",yt[i]);
            }
        }
    }
    return x;
}
```

```
PS D:\dsa\all lab record> cd "d:\dsa\all lab record\" ;
6 8 4 2 1 9
PS D:\dsa\all lab record> cd "d:\dsa\all lab record\" ;
6 8 4 2 1 9
PS D:\dsa\all lab record> cd "d:\dsa\all lab record\" ;

89

78

Finding dulicate complete
PS D:\dsa\all lab record>
```

- **Program to remove duplicate elements in an array**

```c
//Removal of duplicate in array sorted or unsorted works for both case
#include<stdio.h>
#define MAX 6
int duplicate(int yt[]);
void display(int cool[]);
int main()
{
    int ty=0,u=0;
    int arr[MAX]={9,2,7,4,7};
    int f=duplicate(arr);
    u=ty;
    if(f==0)
    {
        printf("\nDuplicate not found in an array deletion failed\n");
    }
    else
    {
        printf("\nFinding and removing duplicate complete\n");
    }
    display(arr);
    return 0;
}
int duplicate(int yt[])
{
    int a=0;
    int k,z=0;
    for(int i=0;i<=MAX-1;i++)
    {
        a=a+1;
        for(int j=a;j<=MAX-1;j++)
        {
            if(yt[i]==yt[j])
```

```
            {
                for(k=j;k<=MAX-1;k++)
                {
                    yt[k]=yt[k+1];
                }
                z=z+1;//counter
            }
        }
    }
    if(z==0)
    {
        return 0;
    }
    else
    {
        return 1;
    }

}
void display(int cool[])
{
    for(int i=0;i<=MAX-1;i++)
    {
        printf("%d ",cool[i]);
    }

}
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights rese

Try the new cross-platform PowerShell https://aka.ms

PS D:\dsa\all lab record> cd "d:\dsa\all lab record\
6 8 4 2 1 9
PS D:\dsa\all lab record>
```

QUESTION NUMBER-2 (2-D ARRAYS)

```
//Program Name:- Sum of two Matrices.
#include<stdio.h>
 void main()
 {
 int a[10][10],b[10][10],c[10][10],i,j,row,col;
```

```
printf("\n Enter the row and column size : ");
scanf("%d %d",&row,&col);

printf("\n Enter the matrix 1 :\n\n");
for(i=0;i<row;i++)
{
for(j=0;j<col;j++)
{
scanf("%d",&a[i][j]);
}
}

printf("\n Enter the matrix 2 :\n\n");
for(i=0;i<row;i++)
{
for(j=0;j<col;j++)
{
scanf("%d",&b[i][j]);
}
}

for(i=0;i<row;i++)
{
for(j=0;j<col;j++)
{
c[i][j]=a[i][j]+b[i][j];
}
}

printf("\n The sum is : \n");
for(i=0;i<row;i++)
{
for(j=0;j<col;j++)
{
printf(" %d ",c[i][j]);
}
printf("\n");
}
}
```

```
 Enter the row and column size : 2
2

 Enter the matrix 1 :

1
2
3
4

 Enter the matrix 2 :

3
4
5
6

 The sum is :
 4  6
 8  10
PS D:\dsa\all lab record>
```

TOWER OF HANOI

```c
//Tower of hanoi
#include<stdio.h>
void TOH(int n,int a,int b,int c);
int main()
{
    int n;
    int a=1,b=2,c=3;
    printf("\nEnter the no of disk\n");
    scanf("%d",&n);
    TOH(n,a,b,c);
    return 1;
}
void TOH(int n,int a,int b,int c)
{
    if(n>0)
    {
        TOH(n-1,a,c,b);
        printf("(%d,%d)",a,c);
        TOH(n-1,b,a,c);
    }
}
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserv

Try the new cross-platform PowerShell https://aka.ms/p

PS D:\dsa\all lab record> cd "d:\dsa\all lab record\"

Enter the no of disk
3
(1,3)(1,2)(3,2)(1,3)(2,1)(2,3)(1,3)
PS D:\dsa\all lab record>
```

QUESTION NUMBER -3(STACKS)

- **Sorting in stack**

```c
//Sorting inside stack
#include<stdio.h>
#define max 100
int s[max],top=-1;
void push(int element);
int pop();
void display();
void sorting();
void insert(int x);
int main()
{
    int data,ch=1,s=1;
    while (s==1)
    {
    printf("enter the choice\n");
    printf("1.push\n");
    printf("2.sorting\n");
    printf("3.display\n");
    printf("4.stop\n");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
        printf("enter the data\n");
        scanf("%d",&data);
        push(data);
        break;
        case 2:
```

```
        //pop();
        sorting();
        display();
        printf("\n");
        break;
        case 3:
        display();
        printf("\n");
        break;
        case 4:
        //exit(1);
        s=0;
        break;
        default:
            printf("wrong choice\n");
        break;
    }
    }
    return 0;
}
void push(int element)
{
    if (top>max-1)
    {
        printf("overflow");
        return;
    }
    else
    {
        top=top+1;
        s[top]=element;
    }
}
int  pop()
{
    int value;
    if (top== -1)
    {
        printf("underflow");
    }
    else
    {
        value=s[top];
        top=top-1;
        return value;
    }
}
void display()
```

```
{
    int i;
    if (top==-1)
    {
        return ;
    }
    i=pop();
    display();
    printf("%d\t",i);
    push(i);

}
void sorting()
{
    if (top!=-1)
    {
        int x=pop();
        sorting();
        insert(x);
    }
}
void insert(int x)
{
    if (top==-1||x>s[top])
    {
        push(x);
        return;
    }
    int t=pop();
    insert(x);
    push(t);
}
```

```
enter the choice
1.push
2.sorting
3.display
4.stop
1
enter the data
56
enter the choice
1.push
2.sorting
3.display
4.stop
1
enter the data
43
enter the choice
1.push
2.sorting
3.display
4.stop
1
enter the data
5
enter the choice
1.push
2.sorting
3.display
4.stop
1
enter the data
34
enter the choice
1.push
2.sorting
3.display
4.stop
2
5       5       34      43      56
```

- ***Infix to Postfix***

```
//INFIX TO POSTFIX

#include<stdio.h>
#define MAX 20
typedef struct{
    int data[MAX];
    int top;
}stack;
 stack s1;
 int push(stack *s,int v)
 {
     if(s->top==MAX-1)
     {
         printf("Overflow");
         return 1;
     }
     else
     {
         s->top++;
         s->data[s->top]=v;
         return 0;
     }
 }
 int pop(stack *s,int *v)
 {
     if(s->top==-1)
     {
         printf("underflow");
         return 1;
     }
     else
     {
         *v=s->data[s->top];
         s->top--;
         return 0;

     }
 }
 int is_operand(int x)
 {
     if((x>=65&& x<=90) || (x>=97 && x<=122) || (x>=48 && x<=57))
     {
         return 1;
     }
     else
     {
```

```
        return 0;
    }
}
int get_v(char op)
{
    int t;
    switch (op)
    {
    case '+':
    case'-':t=1;
        break;
    case'*':
    case'/':t=2;
    break;
    case '^':t=3;
    break;
    default:
    printf("not found sorry");
    }
    return t;
}
int is_l_to_h(char o1,char o2)
{
    if(get_v(o1)<get_v(o2))
    return 1;
    else
        return 0;
}
//operation performed
 int infix_to_postfix(char *inp,char*out)
 {
     int m,k=0,n,i=0;
     while(inp[i]!='\0')
     {
         if(is_operand(inp[i]))
         {
             out[k++]=inp[i];
         }
         else if(inp[i]=='(')
         {
             push(&s1,inp[i]);
         }
         else if(inp[i]==')')
         {
             while(1)
             {
                 pop(&s1,&m);
             if(m=='(')
```

```
                break;
            out[k++]=m;

        }
    }
else
   {
       if(s1.top==-1)
       {
           push(&s1,inp[i]);
       }
       else
       {
           pop(&s1,&n);
           if(n=='(')
           {
               push(&s1,n);
               push(&s1,inp[i]);
           }
           else if(is_l_to_h(n,inp[i]))
               {
                   push(&s1,n);
                   push(&s1,inp[i]);
               }
               else
               {
                   out[k++]=n;
                   continue;
               }
           }
   }
   i++;
   }
           while(s1.top!=-1)
           {
               pop(&s1,&m);
               out[k++]=m;
           }
 }
int main()
{
   char inp[100]; //"a/b-k*(d-e*f+g)/p";
   int l = 0;
   int i = 0;
   printf("Enter infix expression: ");
   gets(inp);
   while(inp[i]!='\0'){
       l++;
```

```
        i++;
    }
    char out[l+1];
    s1.top = -1;
    infix_to_postfix(inp, out);
    i = 0;
    while(out[i]!='\0'){
        printf("%c", out[i]);
        i++;
    }
    return 0;
}
```

```
Enter infix expression: (a+b)+(c*d+t/y)+h
ab+cd*ty/++h+
PS D:\dsa\linking\STACK>
```

- Infix to prefix

```
//Infix to prefix
#include<stdio.h>
#include<math.h>
#include<string.h>
#include <stdlib.h>
#define MAX 20
void push(int);
char pop();
void infix_to_prefix();
int precedence (char);
char stack[20],infix[20],prefix[20];
int top = -1;

int main()
{
printf("\nINPUT THE INFIX EXPRESSION : ");
scanf("%s",infix);
infix_to_prefix();
return 0;
}
void push(int pos)
{
if(top == MAX-1)
{
printf("\nSTACK OVERFLOW\n");
}
else {
```

```
top++;
stack[top] = infix[pos];
}}

char pop()
{
char ch;
if(top < 0)
{
printf("\nSTACK UNDERFLOW\n");
exit(0);
}
else
{
ch = stack[top];
stack[top] = '\0';
top--;
return(ch);
}
return 0;
}
void infix_to_prefix()
{
int i = 0,j = 0;
strrev(infix);
while(infix[i] != '\0')
{
if(infix[i] >= 'a' && infix[i] <= 'z')
{
prefix[j] = infix[i];
j++;
i++;
}
else if(infix[i] == ')' || infix[i] == '}' || infix[i] == ']')
{
push(i);
i++;
}
else if(infix[i] == '(' || infix[i] == '{' || infix[i] == '[')
{
if(infix[i] == '(')
{
while(stack[top] != ')')
{
prefix[j] = pop();
j++;
}
pop();
```

```
i++;
}
else if(infix[i] == '[')
{
while(stack[top] != ']')
{
prefix[j] = pop();
j++;
}
pop();
i++;
}
else if(infix[i] == '{')
{
while(stack[top] != '}')
{
prefix[j] = pop();
j++;
}
pop();
i++;
}}
else
{
if(top == -1)
{
push(i);
i++;
}
else if( precedence(infix[i]) < precedence(stack[top]))
{
prefix[j] = pop();
j++;
while(precedence(stack[top]) > precedence(infix[i])){
prefix[j] = pop();
j++;
if(top < 0) {
break;
}}
push(i);
i++;
}
else if(precedence(infix[i]) >= precedence(stack[top]))
{
push(i);
i++;
}}}
while(top != -1)
```

```
{
prefix[j] = pop();
j++;
}
strrev(prefix);
prefix[j] = '\0';
printf("EQUIVALENT PREFIX NOTATION : %s ",prefix);
}

int precedence(char ci)
{
if( ci== '+' || ci=='-')
{
return(1);
}
if(ci== '*' || ci =='/')
{
return(2);
}
return 0;
}
```

```
INPUT THE INFIX EXPRESSION : a+b*d*(c+d)
EQUIVALENT PREFIX NOTATION : +a**bd+cd
PS D:\dsa\linking\STACK>
```

QUESTION NUMBER-4(QUEUES)

- **Queue using linked list**

```c
//Queue using linked List
#include<stdio.h>
#include<stdlib.h>
#define m 5
struct node
{
    int data;
    struct node *nxt;
};
typedef struct
{
    struct node *r,*f;
}queue;
int insert(queue *q,int n)
{
    struct node *ptr=(struct node*)malloc(sizeof(struct node));
    if(ptr==NULL)
    {
        printf("Full Queue.\n");
        return 1;
    }
    ptr->data=n;
    ptr->nxt=NULL;
    if(q->r==NULL)
    {
        q->r=q->f=ptr;
    }
    else
    {
        q->r->nxt=ptr;
        q->r=ptr;
    }
    return 0;
}
int delete(queue *q,int *n)
{
    if(q->f==NULL)
    {
        printf("Empty Queue.\n");
        return 1;
    }
    if(q->f==q->r)
    {
        *n=q->f->data;
        free(q->f);
```

```
        q->r=q->f=NULL;
    }
    else
    {
        struct node *ptr=q->f;
        *n=q->f->data;
        q->f=q->f->nxt;
        free(ptr);
    }
    return 0;
}
int main()
{
    queue q;
    q.r=q.f=NULL;
    int n,x,y;
    for(int i=0; i<m; i++)
    {
        x=insert(&q,rand()%35);
    }
    for(int i=0; i<m; i++)
    {
        y=delete(&q,&n);
        printf("%d ",n);
    }
    printf("\n");
    return 0;
}
```

```
PS D:\dsa\linking\QUEUE> c
6 22 34 5 24
PS D:\dsa\linking\QUEUE>
```

➢ Circular queue using arrays and Linked list

```c
//Circular queue using array
#include<stdio.h>
#define MAX 5

typedef struct{
    int data[MAX];
    int f;
    int r;
}Cqueue;

int insert(Cqueue *q, int v){
    if(q->f == (q->r+1)%MAX){
        printf("Queue is full\n");
        return 1;
    }
    if(q->r == -1){
        q->f = q->r = 0;
        q->data[q->r] = v;
    }else{
        q->r = (q->r+1)%MAX;
        q->data[q->r] = v;
    }
    return 0;
}

int delete(Cqueue *q, int *m){
    if(q->f == -1){
        printf("Queue is empty\n");
        return 1;
    }
    if(q->f == q->r){
        *m = q->data[q->f];
        q->f = q->r = -1;
    }else{
        *m = q->data[q->f];
        q->f = (q->f+1)%MAX;
    }
    return 0;
}

void display(Cqueue q){
    int i;
    if (q.f == - 1)
        printf("Queue is empty \n");
    else{
        if(q.f <= q.r){
```

```
            i=q.f;
            while(i<=q.r){
                printf("%d ", q.data[i]);
                i++;
            }
        }else{
            i=q.f;
            while(i<MAX){
                printf("%d ", q.data[i]);
                i++;
            }
            i=0;
            while(i<=q.r){
                printf("%d ", q.data[i]);
                i++;
            }
        }
        printf("\n");
    }
}

int main(){
    Cqueue q1;
    q1.f = q1.r = -1;
    int p = insert(&q1, 8);
    p = insert(&q1, 34);
    p = insert(&q1, 2);
    p = insert(&q1, 78);
    p = insert(&q1, 11);
    display(q1);
    int m;
    int q = delete(&q1, &m);
    display(q1);
    p = insert(&q1, 9);
    display(q1);
    return 0;
}
```

```
8 34 2 78 11
34 2 78 11
34 2 78 11 9
PS D:\dsa\linking\QUEUE>
```

- **Circular queue using linked list**

```c
#include<stdio.h>
#include<stdlib.h>
#define m 5
struct node
{
    int data;
    struct node *nxt;
};
typedef struct
{
    struct node *r;
}queue;
int insert(queue *q, int n)
{
    struct node *ptr=(struct node*)malloc(sizeof(struct node));
    if(ptr==NULL)
    {
        printf("Insetion Not Possible.\n");
        return 1;
    }
    ptr->data=n;
    ptr->nxt=NULL;
    if(q->r==NULL)
    {
        q->r=ptr;
        q->r->nxt=q->r;
    }
    else
    {
        ptr->nxt=q->r->nxt;
        q->r->nxt=ptr;
        q->r=ptr;
    }
    return 0;
}
int delete(queue *q, int *n)
{
    if(q->r==NULL)
    {
        printf("Deletion Not Possible.\n");
        return 1;
    }
    if(q->r==q->r->nxt)
    {
        *n=q->r->data;
```

```
        free(q->r);
        q->r=NULL;
    }
    else
    {
        struct node *ptr=q->r->nxt;
        q->r->nxt=ptr->nxt;
        *n=ptr->data;
        free(ptr);
    }
    return 0;
}
int main()
{
    queue q;
    q.r=NULL;
    int n,x,y;
    for(int i=0; i<m;i++)
    {
        scanf("%d",&n);
        x=insert(&q,n);
    }
    y=delete(&q,&n);
    printf("%d ",n);
    y=delete(&q,&n);
    printf("%d ",n);
    printf("\n");
    return 0
}
```

```
34
21
78
3
21
34 21
PS D:\dsa\linking\QUEUE>
```

➢ Implement stack using a queue

```c
//Queue implementation using stack
#include<stdio.h>
#include<stdlib.h>
#define MAX 20

typedef struct{
    int data[MAX];
    int front;
    int rear;
}Queue;

Queue q1;

int insert(Queue *q, int v){
    if(q->rear == MAX-1){    //full Q
        printf("Queue is full\n");
        return 1;
    }
    if(q->rear == -1){   //Empty Q
        q->front = q->rear = 0;
        q->data[q->rear] = v;
    }else{   //Partially full Q
        q->rear++;
        q->data[q->rear] = v;
    }
    return 0;
}

int delete(Queue *q, int *m){
    if(q->front == -1){   //Empty Q
        printf("Q is empty\n");
        return 1;
    }
    if(q->front == q->rear){
        *m = q->data[q->front];
        q->front = q->rear = -1;
    }else{
        *m = q->data[q->front];
        q->front++;
    }
    return 0;
}

int delete2(Queue *q, int *m){
    if(q->front == -1){
        printf("Q is empty\n");
```

```
        return 1;
    }
    if(q->front == q->rear){
        *m = q->data[q->front];
        q->front = q->rear = -1;
    }else{
        *m = q->data[q->front];
        for(int i=1; i<q->rear; i++){
            q->data[i-1] = q->data[i];
        }
        q->rear--;
    }
    return 0;
}

void display(Queue q){
    int i;
    if (q.front == - 1)
        printf("Queue is empty \n");
    else{
        for (i = q.front; i <= q.rear; i++)
            printf("%d ", q.data[i]);
        printf("\n");
    }
}

int isEmpty(Queue q){
    return (q.front == -1) ? 1 : 0;
}

int push(int v){
    return insert(&q1, v);
}

int pop(int *m){
    int p = isEmpty(q1);
    if(p) return p;
    int i = q1.front;
    int j = q1.rear;
    while(i!=j){
        int n;
        delete(&q1, &n);
        insert(&q1, n);
        i++;
    }
    int n;
    delete(&q1, &n);
    *m = n;
```

```
    return 0;
}

int main(){
    q1.front = q1.rear = -1;
    int t = push(60);
    display(q1);
    t = push(60);
    display(q1);
    t = push(120);
    display(q1);
    int m;
    int r = pop(&m);
    display(q1);
    return 0;
}
```

```
PS D:\dsa\linking\QUEUE>
60
60 60
60 60 120
60 60
PS D:\dsa\linking\QUEUE>
```

➢ Implement queue using a stack

```
//Stack using queue
#include<stdio.h>
#include<stdlib.h>
#define MAX 20

typedef struct{
    int data[MAX];
    int top;
}STACK;

STACK st[2];

int push(STACK *S, int v){
    if(S->top == MAX-1){
        printf("Overflow\n");
        return 1;
    }
```

```
    S->top++;
    S->data[S->top] = v;
    return 0;
}

int pop(STACK *S, int *v){
    if(S->top == -1){
        printf("Underflow\n");
        return 1;
    }
    *v = S->data[S->top];
    S->top--;
    return 0;
}

void display(STACK *S){//using recursion
    if(S->top == -1) return;
    int u;
    pop(S, &u);
    printf("%d ", u);
    display(S);
    push(S, u);
}

void display_rev(STACK *S){
    if(S->top == -1) return;
    int u;
    pop(S, &u);
    display_rev(S);
    printf("%d ", u);
    push(S,u);
}

int isEmpty(STACK S){
    return (S.top == -1) ? 1 : 0;
}

int enqueue(int v){
    return push(&st[0], v);
}

int dequeue(int *m){
    int p = isEmpty(st[0]);
    if(p) return p;
    p = isEmpty(st[1]);
    if(p == 1){
        int n;
        while(st[0].top != -1){
```

```
            pop(&st[0], &n);
            push(&st[1], n);
        }
        pop(&st[1], &n);
        *m = n;
        return 0;
    }
    int n;
    pop(&st[1], &n);
    *m = n;
    return 0;
}

int main(){
    st[0].top = st[1].top = -1;
    int t = enqueue(20);
    display_rev(&st[1]);
    display(&st[0]);
    printf("\n");
    t = enqueue(40);
    display_rev(&st[1]);
    display(&st[0]);
    printf("\n");
    t = enqueue(80);
    display_rev(&st[1]);
    display(&st[0]);
    printf("\n");
    int m;
    int r = dequeue(&m);
    display_rev(&st[1]);
    display(&st[0]);
    printf("\n");
    return 0;
}
```

```
20
40 20
80 40 20
80 40
PS D:\dsa\linking\QUEUE>
```

QUESTION NUMBER-5 (LINKED LIST)

- Remove duplicates from a linked list

```c
//Removal of duplicate in linked list
//Removal of duplicate in linked list
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
}*head=NULL;

void create(struct node *p,int a[])
{
    struct node *curr,*ptr;
    for(int i=0;i<7;i++)
    {
        curr=(struct node *)malloc(sizeof(struct node *));
        curr->data=a[i];
        curr->next=NULL;
        if(head==NULL)
        {
            head=curr;
            ptr=curr;
        }
        else
        {
            ptr->next=curr;
            ptr=curr;
        }

    }
}
void duplicate(struct node *p)
{
    struct node *q;
    struct node *r=p;
    while(p!=NULL)
    {
        q=p->next;
        while(q!=NULL)
        {
        if(p->data==q->data)
        {
            r->next=q->next;
            free(q);
```

```
            q=r->next;
        }
        else
        {
            r=q;
            q=q->next;
        }
        }
        p=p->next;

    }
}
void display(struct node *q)
{
    while(q!=NULL)
    {
        printf("%d ",q->data);
        q=q->next;
    }
}
int main()
{
    int arr[7]={6,8,4,6,2,8,9};
    create(head,arr);
    duplicate(head);
    display(head);
    return 0;
}
```

```
PS D:\dsa\linking\set of linked list> c
duplicatedeletion }
6 8 4 2 9
PS D:\dsa\linking\set of linked list\si
```

- Reverse a linked list

```
//Reversing a linked list
//Removal of duplicate in linked list
#include<stdio.h>
#include<stdlib.h>

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

struct node *start = NULL;

struct node *create(struct node *start){
    int n;
    struct node * new_node = (struct node *)malloc(sizeof(struct node));
    printf("enter value  : ");
    scanf("%d",&n);
    new_node->data =n;
    new_node->next =start;
    start=new_node;
    return start;
}


struct node *ins_end(struct node* start){
    int n;
    struct node *new_node = (struct node *)malloc(sizeof(struct node));
    struct node *ptr;
    printf("enter value: ");
    scanf("%d",&n);
    new_node->data =n;
    new_node->next =NULL;
    ptr = start;
    while(ptr->next !=NULL){
        ptr =ptr->next;
    }
    ptr->next= new_node;
    return start;
}

struct node *rev(struct node*start){

struct node *ptr,*preptr=NULL,*temp;
    ptr=start;
    while(ptr!=NULL){
        temp=ptr->next;
        ptr->next=preptr;
```

```
        preptr =ptr;
        ptr=temp;
    }
    start= preptr;
    return start;
}


void display(struct node *start){
    struct node *temp;
    if(start == NULL)
        printf("the linked list doesn't exists. ");
    else{
        temp = start;
        while(temp != NULL){
            printf("  %d",temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
}




int main(){
    int ch;
    while(1){
        printf("1.   Create .\n");
        printf("2.   Insert at end.\n");
        printf("3.   reverse the entered link list.\n");
        printf("4.   Display .\n");
        printf("5.   Exit .\n");
        printf(" Enter your choice : ");
        scanf("%d",&ch);
        switch(ch){
            case 1:
                start = create(start);
                break;
            case 2:
                start = ins_end(start);
                break;
            case 3:
                start = rev(start);
                break;
            case 4:
                display(start);
                break;
            case 5:
```

```
                exit(0);
                break;
            default:
                printf(" wrong choice . \n");
        }
    }
    return 0;
}
```

```
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 1
enter value  : 45
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 2
enter value: 56
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 2
enter value: 89
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 3
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
 Enter your choice : 4
  89  56  45
1.   Create .
2.   Insert at end.
3.   reverse the entered link list.
4.   Display .
5.   Exit .
```

QUESTION NUMBER - 6(TREES)

- Height of a binary tree

```c
//Height of tree
#include<stdio.h>
#include<stdlib.h>
struct node {
    int data;
    struct node *lptr;
    struct node *rptr;
};
void create(struct node **root,int data);
void *display(struct node *NODE);
int  height(struct node *root);
int main()
{
    int ch ,dat;
    struct node *root=NULL;
    struct node *NODE=NULL;
    while (1)
    {
        printf("1.create\n");
        printf("2.display\n");
        printf("3.exit\n");
        printf("4.height of the tree\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("enter the data \n");
                scanf("%d",&dat);
                create(&root,dat);
                break;
            case 2:
                NODE=root;
                display(NODE);
                printf("\n");
                break;
            case 3:
                exit(1);
            case 4:
                printf("the height of the tree is %d\n",height(root));
                break;
            default:
                printf("wrong choice\n");
        }
    }
```

```
    return 0;
}
void create(struct node **root,int data)
{
    struct node *newnode,*parent,*ptr;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=data;
    newnode->lptr=NULL;
    newnode->rptr=NULL;
    if (*root==NULL)
    {
        *root=newnode;
    }
    else
    {
        ptr=*root;
        parent=*root;
        while(ptr!=NULL)
        {
            parent=ptr;
            if (data<ptr->data)
            {
                ptr=ptr->lptr;
            }
            else
            {
                ptr=ptr->rptr;
            }
        }
    if (data<parent->data)
    {
        parent->lptr=newnode;
    }
    else
    {
        parent->rptr=newnode;
    }
  }
}
void *display(struct node *NODE)
{
//  preorder print
    if (NODE!=NULL)
    {
        printf("\n%d",NODE->data);
        display(NODE->lptr);
        display(NODE->rptr);
    }
```

```
}
int height(struct node *root)
{
    if (root==NULL)
       return 0;
   else
   {
       int left = height(root->lptr);
       int right = height(root->rptr);
       if (left > right)
           return(left+1);
       else return(right+1);
   }
}
```

```
1.create
2.display
3.exit
4.height of the tree
1
enter the data
56
1.create
2.display
3.exit
4.height of the tree
1
enter the data
67
1.create
2.display
3.exit
4.height of the tree
1
enter the data
32
1.create
2.display
3.exit
4.height of the tree
1
enter the data
90
1.create
2.display
3.exit
4.height of the tree
4
the height of the tree is 3
1.create
2.display
3.exit
4.height of the tree
```

➢ Find kth maximum value in a binary search tree

```c
//KTH maximum value in a binary search tree
#include <stdio.h>

struct Node {
    int data;
    struct Node *left, *right;
};

struct Node* newNode(int data)
{
    struct Node* temp = malloc(sizeof(struct Node));
    temp->data = data;
    temp->right = temp->left = NULL;
    return temp;
}

struct Node* KthLargestUsingMorrisTraversal(struct Node* root, int k)
{
    struct Node* curr = root;
    struct Node* Klargest = NULL;

    int count = 0;

    while (curr != NULL) {
        if (curr->right == NULL) {

            if (++count == k)
                Klargest = curr;

            curr = curr->left;
        }

        else {

            struct Node* succ = curr->right;

            while (succ->left != NULL && succ->left != curr)
                succ = succ->left;

            if (succ->left == NULL) {

                succ->left = curr;
```

```
                curr = curr->right;
            }

            else {

                succ->left = NULL;

                if (++count == k)
                    Klargest = curr;

                curr = curr->left;
            }
        }
    }

    return Klargest;
}

int main()
{
    /* Constructed binary tree is
          4
        /   \
       2     7
     /  \   /  \
    1    3  6    10 */

    struct Node* root = newNode(4);
    root->left = newNode(2);
    root->right = newNode(7);
    root->left->left = newNode(1);
    root->left->right = newNode(3);
    root->right->left = newNode(6);
    root->right->right = newNode(10);

    printf("Finding K-
th largest Node in BST : %d\n", KthLargestUsingMorrisTraversal(root, 3)->data);

    return 0;
}
```

```
    2 |
Finding K-th largest Node in BST : 6
PS D:\dsa\all lab record>
```

QUESTION NUMBER -7 (SEARCHING AND SORTING)

- Bubble Sort

```c
//Bubble sort
#include <stdio.h>
 int main()
 {
 int array[100], n, c, d, swap;
 printf("Enter number of elements\n");
 scanf("%d", &n);
 printf("Enter %d integers\n", n);
 for (c = 0; c < n; c++)
 scanf("%d", &array[c]);
 for (c = 0 ; c < n - 1; c++)
 {
 for (d = 0 ; d < n - c - 1; d++)
 {
 if (array[d] > array[d+1])
 {
 swap = array[d];
 array[d] = array[d+1];
 array[d+1] = swap;
 }
 }
 }
 printf("Sorted list in ascending order:\n");
 for (c = 0; c < n; c++)
 printf("%d\n", array[c]);
 return 0;
 }
```

```
Enter number of elements
5
Enter 5 integers
23
45
12
54
23
Sorted list in ascending order:
12
23
23
45
54
PS D:\dsa\linking\Search>
```

- Insertion Sort

```c
//Insertion sort
#include <stdio.h>
 int main()
 {
 int n, array[1000], c, d, t;
 printf("Enter number of elements\n");
 scanf("%d", &n);
 printf("Enter %d integers\n", n);
 for (c = 0; c < n; c++)
 scanf("%d", &array[c]);
 for (c = 1 ; c <= n - 1; c++) {
 d = c;
 while ( d > 0 && array[d-1] > array[d]) {
 t = array[d];
 array[d] = array[d-1];
 array[d-1] = t;
 d--;
 }
 }
 printf("Sorted list in ascending order:\n");
 for (c = 0; c <= n - 1; c++) {
 printf("%d\n", array[c]);
 }
 return 0;
 }
```

```
Enter number of elements
4
Enter 4 integers
57
89
43
1
Sorted list in ascending order:
1
43
57
89
PS D:\dsa\linking\Search>
```

➢ Merge Sort

```
//Merge sort
#include <stdio.h>
 #define max 10
 int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
 int b[10];
 void merging(int low, int mid, int high) {
 int l1, l2, i;
 for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
 if(a[l1] <= a[l2])
 b[i] = a[l1++];
 else
 b[i] = a[l2++];
 }
 while(l1 <= mid)
 b[i++] = a[l1++];

 while(l2 <= high)
 b[i++] = a[l2++];

 for(i = low; i <= high; i++)
 a[i] = b[i];
 }
 void sort(int low, int high) {
 int mid;
 if(low < high) {
 mid = (low + high) / 2;
 sort(low, mid);
 sort(mid+1, high);
 merging(low, mid, high);
 } else {
 return;
 }
 }

 int main() {
 int i;
 printf("List before sorting\n");
 for(i = 0; i <= max; i++)
 printf("%d ", a[i]);
 sort(0, max);
 printf("\nList after sorting\n");
 for(i = 0; i <= max; i++)
 printf("%d ", a[i]);
 }
```

```
PS D:\dsa\linking\Search> cd "d:\dsa\li
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
PS D:\dsa\linking\Search> []
```

- Quick Sort

```
//Quick sort
#include<stdio.h>
 void quicksort(int number[25],int first,int last){
 int i, j, pivot, temp;
 if(first<last){
 pivot=first;
 i=first;
 j=last;
 while(i<j){
 while(number[i]<=number[pivot]&&i<last)
 i++;
 while(number[j]>number[pivot])
 j--;
 if(i<j){
 temp=number[i];
 number[i]=number[j];
 number[j]=temp;
 }
 }
 temp=number[pivot];
 number[pivot]=number[j];
 number[j]=temp;
 quicksort(number,first,j-1);
 quicksort(number,j+1,last);
 }
 }

 int main(){
 int i, count, number[25];
 printf("How many elements are u going to enter?: ");
 scanf("%d",&count);
 printf("Enter %d elements: ", count);
 for(i=0;i<count;i++)
 scanf("%d",&number[i]);
 quicksort(number,0,count-1);
 printf("Order of Sorted elements: ");
 for(i=0;i<count;i++)
```

```
printf(" %d",number[i]);

return 0;
}
```

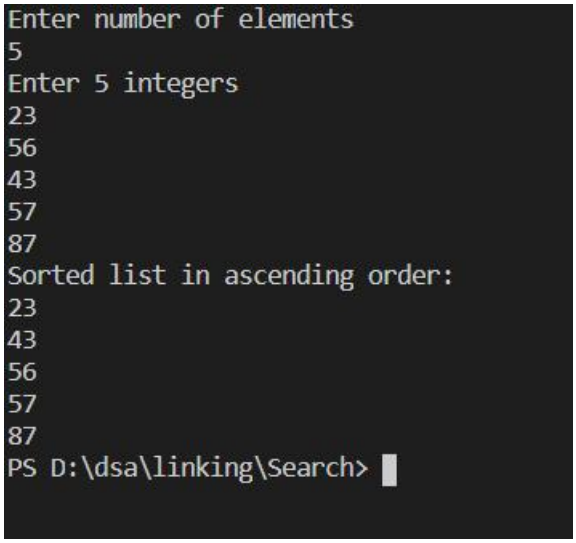
```
How many elements are u going to enter?: 5
Enter 5 elements: 35
56
90
67
24
Order of Sorted elements:  24 35 56 67 90
```

- Selection Sort

```
//Selection sort
#include <stdio.h>
 int main()
 {
 int array[100], n, c, d, position, swap;
 printf("Enter number of elements\n");
 scanf("%d", &n);
 printf("Enter %d integers\n", n);
 for (c = 0; c < n; c++)
 scanf("%d", &array[c]);
 for (c = 0; c < (n - 1); c++)
 {
 position = c;
 for (d = c + 1; d < n; d++)
 {
 if (array[position] > array[d])
 position = d;
 }
 if (position != c)
 {
 swap = array[c];
 array[c] = array[position];
 array[position] = swap;
 }
 }

 printf("Sorted list in ascending order:\n");

 for (c = 0; c < n; c++)
 printf("%d\n", array[c]);
 return 0;
 }
```



2020 LAB RECORD

DSA

KIIT UNIVERSITY

ROLL NO-1905995