C program for Single Linked List

```
#include<stdio.h>
#include<stdlib.h>
struct node {
  int data;
  struct node *next;
};
insert(struct node **p,int num)
{
        struct node *temp=NULL;
        struct node *current=*p;
        temp=(struct node *)malloc( sizeof(struct node));
        temp->data=num;
        temp->next=NULL;
        if(*p==NULL)
        {
                *p=temp;
                return;
        }
        while(current->next!= NULL)
                current=current->next;

        current->next=temp;

}
void add_first(struct node **p)
{
        struct node *temp=NULL;
        int num;
        printf("\nEnter the number:");
        scanf("%d",&num);
        temp=(struct node*)malloc(sizeof(struct node));
        temp->data=num;
        if(*p==NULL)
                temp->next=NULL;
        else
                temp->next=*p;
        *p=temp;
}

void add_after(struct node *p)
{
        struct node *temp=NULL;
        struct node *current;
        int pos,num;
        current=p;
        printf("\nenter the position:");
        scanf("%d",&pos);
```

```
        while((pos>1) && (current!=NULL))
        {
                current=current->next;
                pos--;
        }
        if(current==NULL)
        {
                printf("\nThis position is not found in the list.");
                return;
        }
        printf("\nEnter the number you want to insert:");
        scanf("%d",&num);
        temp=(struct node*)malloc(sizeof(struct node));
        temp->data=num;
        temp->next=current->next;
        current->next=temp;
}

int del_node(struct node **p)
{
        struct node *temp,*prev;
        int num;
        temp=*p;
        printf("\nEnter the number you want to delete:");
        scanf("%d",&num);
        while(temp!=NULL)
        {
                if(temp->data==num)
                {
                        if(temp==*p)
                                *p=temp->next;
                        else
                                prev->next=temp->next;
                        free(temp);
                        return;
                }
                prev=temp;
                temp=temp->next;
        }

}

void count(struct node *p)
{
        int c=0;
        while(p!=NULL)
        {
```

```
                c++;
                p=p->next;
        }
        printf("\ntotal node count=%d",c);
}

void display(struct node *p)
{
        printf("\nThe final list is showing below:\n");
        while(p!=NULL)
        {
                printf("\t%d",p->data);
                p=p->next;
        }
}


main()
{
        struct node *q=NULL;


  insert(&q,10);
display(q);
  insert(&q,20);
display(q);
  insert(&q,30);
display(q);
  insert(&q,40);
display(q);
  insert(&q,50);
display(q);
  insert(&q,60);
display(q);
  insert(&q,70);
display(q);
  insert(&q,80);
display(q);
  insert(&q,90);
display(q);
  insert(&q,100);
display(q);

count(q);

add_first(&q);
display(q);
```

```
count(q);

add_after(q);
display(q);
add_after(q);
display(q);
add_after(q);
display(q);

count(q);


del_node(&q);
display(q);
count(q);
del_node(&q);
display(q);


}
```

QUEUE IMPLEMENTATION USING ARRAY

```
# include<stdio.h>

# define size 10

int rear, front;
int ch;
int q[size];

int rear = -1;
int front = -1;
void Insert_queue();
void Delete_queue();
void Display_queue();

/* Function to create queue */

void  Insert_queue()
{
        printf("\n Input the element :");
        scanf("%d", &ch);
        if(rear < size-1)
        {
                rear ++;
```

```
                q[rear] = ch ;
                if(front == -1)
                        front = 0;
        }
        else
                printf("\n Queue is overflow");
}

/* Function to perform delete operation */

void Delete_queue()
{
        if (front == -1)
        {
                printf("\n Underflow");
                return ;
        }
        else
        {
                ch = q[front];

                printf("\n Element deleted : %d", ch);
        }
        if(front == rear)
        {
                front = -1;
                rear = -1;
        }
        else
                front = front + 1;
}




/* Output function */

void Display_queue()  //char q[])
{
        int i;
        if (front == 0)
                return;
        for(i = front ; i <= rear; i++)
                printf(" %d ", q[i]);
}
```

```
/* Function main */


void main()
{
        int a;
        char choice;


        do
        {
                printf("\nInsert->1\n Delete->2 \n Quit->3");
                printf("\nInput the choice : ");
                scanf("%d",&choice);

                switch(choice)
                {
                case 1 :

                        Insert_queue();
                        printf("\nQueue after inserting ");
                        Display_queue();
                        break;

                case 2 :
                        Delete_queue();
                        printf("\nQueue content after deleteion is as follows:\n");
                        Display_queue();
                        break;

                default :
                        exit(0);
                }
        } while(1);
}
```

QUEUE IMPLEMENTATION USING LINKED LIST

```
#include<stdlib.h>
#include<stdio.h>
struct node {
  int data;
  struct node *next;
};

struct queue
{
   struct node *front, *rear;
};

void enqueue(struct queue *q,int num)
{
        struct node *temp;
        temp=(struct node *)malloc(sizeof(struct node));
        temp->data=num;
        temp->next=NULL;
        if(q->rear==NULL && q->front==NULL)
        {
                q->rear=q->front=temp;
                return;
        }

                q->rear->next=temp;
                q->rear=temp;

}




int dequeue(struct queue *q)
{
        struct node *temp;
        int n;
        n=q->front->data;
        temp=q->front;

        if(q->front==q->rear)
        {
                printf("\nunderflow..");
                q->front=q->rear=NULL;
        }
        else
```

```
        {

        q->front=q->front->next;
   }
        free(temp);
        printf("\nThe value deleted from the Queue is %d",n);
}









void main()
{
        int a;
        char choice;
        struct queue *q;
        q=(struct queue*)malloc(sizeof(struct queue));
        q->front=NULL;
        q->rear=NULL;

        do
        {
                printf("\nInsert->1\n Delete->2 \n Quit->3");
                printf("\nInput the choice : ");
                scanf("%d",&choice);

                switch(choice)
                {
                case 1 :
                        printf("\nEnter any number:");
                        scanf("%d",&a);
                        enqueue(q,a);
                        printf("\n%d is inserted.",a);
                        break;

                case 2 :
                        dequeue(q);
```

```
                        break;

                default :
                        exit(0);
                }
        } while(1);
}
```

STACK IMPLEMENTATION USING ARRAY

```
# include<stdio.h>
# include<string.h>
# define size  100

int top = -1;
int stack[size];

void push(int *, int);
int pop(int *);
void display(int *);


void push(int s[], int d)
{
        if(top ==(size-1))
                printf("Stack is overflow.can not able to push.");
        else
        {
                ++top;
                s[top] = d;
        }
}


int pop(int s[])
```

```
{
        int popped_element;
        if(top == -1)
                printf("Stack is underflow.");
        else
        {
                popped_element = s[top];
                --top;
        }
        return (popped_element);
}



void display(int s[])
{
        int i;
        if(top == -1)
        {
                printf("\n Stack is empty");
        }
        else
        {
                for(i = top; i >= 0; --i)
                        printf("\n %d", s[i] );
        }
}









void main()
{
        int  data;
        char choice;
        int q = 0;
        int top = -1;
        do
```

```
	{
		printf(" \nPush->i Pop->p Quit->q:");
		printf("\nInput the choice : ");
		do
		{
			choice = getchar();
			choice =tolower(choice);
		}while(strchr("ipq",choice)==NULL);
		printf("Your choice is: %c",choice);
		switch(choice)
		{
		case 'i' :
			printf("\n Input the element to push:");
			scanf("%d", &data);
			push(stack, data);
			printf("\n After inserting ");
			display(stack);
			if(top == (size-1))
				printf("\n Stack is full");
			break;
		case 'p' :
			data = pop(stack);
				printf("\n Data is popped: %d", data);
				printf("\n Rest data in stack is as follows:\n");
				display(stack);
				break;
		case 'q':
			q = 1;
		}
	} while(!q);
}
```

STACK IMPLEMENTATION USING LINKED LIST

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

struct link
{
        int info;
        struct link *next;
} *start;


void display(struct link *);
struct link *push(struct link *);
struct link *pop(struct link *);
int main_menu();

void display(struct link *rec)
{
        while(rec != NULL)
        {
                printf(" %d ",rec->info);
                rec = rec->next;
        }
}

struct link * push(struct link *rec)
{
        struct link *new_rec;
        printf("\n Input the new value for next location of the stack:");

        new_rec = (struct link *)malloc(sizeof(struct link));
        scanf("%d", &new_rec->info);
        new_rec->next = rec;
        rec = new_rec;
        return(rec);
}

struct  link * pop(struct link *rec)
{
        struct link *temp;

        if(rec == NULL)
        {
                printf("\n Stack is empty");
        }
```

```
        else
        {
                temp = rec->next;
                free(rec);
                rec = temp;
                printf("\n After pop operation the stack is as follows:\n");
                display(rec);
                if(rec  == NULL)
                        printf("\n Stack is empty");
        }
        return(rec);
}




int main_menu ()
{
        int choice;
        do
        {
                printf("\n 1<-Push ");
                printf("\n 2<-Pop");
                printf("\n 3<-Quit");
                printf("\n Input your choice :");
                scanf("%d", &choice);
                if(choice <1 || choice >3)
                        printf("\n Incorrect choice-> Try once again");
        } while(choice <1 || choice >3);

        return(choice);
}

void main()
{
        struct link *start ;
        int choice;
        start = NULL;
        do
        {
                choice = main_menu();
                switch(choice)
                {
                case 1:
```

```
                start = push(start);
                printf("\n After push operation stack is as follows:\n");
                display(start);
                break;
        case 2:
                start = pop(start);
                break;
        default :
                printf("\n End of session");
                exit(0);
        }
    } while(choice != 3);
}
```

Recursion

```
#include<stdio.h>

long int factorial(int n);

int main() {

   int n;

   printf("Enter a positive integer: ");

   scanf("%d",&n);

   printf("Factorial of %d = %ld", n, factorial(n));

   return 0;

}


long int factorial(int n)

{

   if (n>=1)

      return n*factorial(n-1);
```

```
    else
        return 1;
}
```

Bubble Sort

```
#include<stdio.h>
main()
{
        int input[10];
        int i,j,n,temp;
        printf("Enter how many number you want to input:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nEnter number:");
                scanf("%d",&input[i]);
        }
        for(i=1;i<n;i++)
        {
                for(j=0;j<n-i;j++)
                {
                        if(input[j]>input[j+1])
                        {
```

```
                                temp=input[j];
                                input[j]=input[j+1];
                                input[j+1]=temp;
                        }
                }
        }
        printf("The sorted array are as follows:\n");
        for(i=0;i<n;i++)
        printf("%d\t",input[i]);
}
```

Linear Search

```
#include<stdio.h>
main()
{
        int input[10];
        int i,n,search,position=1;
        printf("Enter how many number you want to input:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\nEnter number:");
                scanf("%d",&input[i]);
        }
```

```
	printf("Enter which number you want to search:");
	scanf("%d",&search);
	for(i=0;i<10;i++)
	{
		if(input[i]==search)
		{
			printf("%d is found in %d position.",search,position);
			return;
		}
		position++;

	}
	printf("The search number %d is not available in the list.");
}
```

Binary Search

```
#include<stdio.h>
int binarysearch(int arr[],int x,int low,int high)
{
	int mid;
	while(low<=high)
	{
```

```
		mid=(low+high)/2;
		if(arr[mid]==x)
			return ++mid;
		if(arr[mid]>x)
			high=mid-1;
		else
			low=mid+1;
	}
	return (-1);
}
main()
{
	int input[10];
	int i,j,n,temp,result,search;
	printf("Enter how many number you want to input:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter number:");
		scanf("%d",&input[i]);
	}
	printf("\nEnter the number you want to search");
	scanf("%d",&search);
	for(i=1;i<n;i++)
	{
		for(j=0;j<n-i;j++)
```

```
            {
                if(input[j]>input[j+1])
                {
                    temp=input[j];
                    input[j]=input[j+1];
                    input[j+1]=temp;
                }
            }
        }
        printf("The sorted array are as follows:\n");
        for(i=0;i<n;i++)
        printf("%d\t",input[i]);
        result=binarysearch(input,search,0,n);
        if(result==-1)
        printf("\n%d not found in the list.",search);
        else
        printf("\n%d is found at position %d",search,result);
}
```