1. Stack using Array

```
#include<stdio.h>
#include<conio.h>
#define size 3
int top=-1;
void push(int stk[],int val);
int pop(int stk[]);
void display(int stk[]);
int peak(int stk[]);
void main()
{
        int stack[10],value,ch,i;
        clrscr();
        do
        {
                printf("\n**********Main Menu**********\n");
                printf("1-Push\n2-Pop\n3-Display\n4-Peep\n**************************************\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                printf("Enter the number to be pushed to stack\n");
                                scanf("%d",&value);
                                push(stack,value);
                                break;
                        case 2:
                                value=pop(stack);
                                printf("Delete item=%d\n",value);
                                break;
                        case 3:
                                display(stack);
                                break;
                        case 4:
```

```
                                        value=peak(stack);
                                        printf("Value at the top of stack= %d",value);
                                        break;
                                default:
                                        printf("Invalid Choice");
                        }
        }while(ch<=4&&ch>=1);
        getch();
}
void push(int stk[],int val)
{
        if(top==size-1)
        {
                printf("Stack is full\n");
                return;
        }
        else
        {
                top++;
                stk[top]=val;
        }
}
int pop(int stk[])
{
        int val;
        if(top==-1)
        {
                printf("The stack is empty\n");
                return -1;
        }
        else
        {
                val=stk[top];
                top--;
                return val;
```

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

2. Stack using Linked List

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct stack
{
        int data;
        struct stack *next;
};
struct stack *top=NULL;
struct stack *push(struct stack *,int);
struct stack *display(struct stack *);
struct stack *pop(struct stack *);
int peek(struct stack *);
int main()
{
        int val,ch;
        //clrscr();
        do
        {
                printf("\n*************Main Menu***************\n");
                printf("1.Push\n2.Pop\n3.Peek\n4.Display\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                printf("Enter the element to the stack\n");
                                scanf("%d",&val);
                                top=push(top,val);
                                break;
                        case 2:
                                top=pop(top);
                                break;
```

```
                        case 3:
                                val=peek(top);
                                if(val!=1)
                                        printf("The value at top of stack is %d\n",val);
                                else
                                        printf("Stack is empty\n");
                                break;
                        case 4:
                                top=display(top);
                                break;
                }
        }while(ch>=1&&ch<=4);
        getch();
        return 0;
}
struct stack *push(struct stack *top, int val)
{
        struct stack *ptr;
        ptr=(struct stack *)malloc(sizeof(struct stack *));
        ptr->data=val;
        ptr->next=NULL;
        if(top==NULL)
        {
                top=ptr;
        }
        else
        {       ptr->next=top;
                top=ptr;
        }
        return top;
}
struct stack *pop(struct stack *top)
{
        struct stack *ptr;
        ptr=top;
```

```
	if(top==NULL)
		printf("Stack is Overflow\n");
	else
	{
		top=top->next;
		printf("The value being deleted is %d\n",ptr->data);
		free(ptr);
	}
	return top;
}
int peek(struct stack *top)
{
	if(top==NULL)
		return -1;
	else
		return top->data;
}
struct stack *display(struct stack *top)
{
	struct stack *ptr;
	ptr=top;
	if(top==NULL)
		printf("Stack is empty\n");
	else
	{
		while(ptr!=NULL)
		{
			printf("%d\t",ptr->data);
			ptr=ptr->next;
		}
	}
	return top;
}
```

3. Queue using Array

```
#include<stdio.h>
#include<conio.h>
#define size 3
int front=-1,rear=-1;
void insert(int queue[],int val);
void delete(int queue[]);
void display(int queue[]);
int peak(int queue[]);
void main()
{
        int queue[size],value,ch,i;
        clrscr();
        do
        {
                printf("\n**********Main Menu**********\n");
                printf("1-Insert\n2-Delete\n3-Display\n4-Peak\n**************************************************\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                printf("Enter the number to be insert to queue\n");
                                scanf("%d",&value);
                                insert(queue,value);
                                break;
                        case 2:
                                delete(queue);
                                break;
                        case 3:
                                display(queue);
                                break;
                        case 4:
                                value=peak(queue);
```

```
                printf("Value at the rear of queue= %d",value);
                break;
            default:
                printf("Invalid Choice");
        }
    }while(ch<=4&&ch>=1);
    getch();
}
void insert(int queue[],int val)
{
    if(rear==size-1)
    {
        printf("Queue is full\n");
        return;
    }
    else if(front==-1&&rear==-1)
      {
        front=0;
        rear=0;
      }
      else
            rear++;
    queue[rear]=val;
}
void delete(int queue[])
{
    int val;
    if((front==-1)||(front>rear))
    {
        printf("The queue is empty\n");
    }
    else
    {
        val=queue[front];
        front++;
```

```
                printf("Deleted item=%d",val);
        }
}
void display(int queue[])
{
        int i;
        if((front==-1)||(front>rear))
        {
                printf("Queue is empty\n");
                return;
        }
        else
        {
                for(i=front;i<=rear;i++)
                {
                        printf("%d\t",queue[i]);
                }
        }
}
int peak(int queue[])
{
        int i;
        if((front==-1)&&(rear==-1))
        {
                printf("Queue is empty\n");
                return -1;
        }
        else
                return (queue[rear]);
}
```

4. Queue using Linked List

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node
{
        int data;
        struct node *next;
};
struct queue
{
        struct node *front;
        struct node *rear;
};
struct queue *q;
void create_queue(struct queue *);
struct queue *insert(struct queue *,int);
struct queue *delete_element(struct queue *);
struct queue *display(struct queue *);
int peek(struct queue *);
int main()
{
        int val,ch;
        //clrscr();
        do
        {
                printf("\n***************Mian Menu*************\n");
                printf("1.Insert\n2.Delete\n3.Peek\n4.Display\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:
                                printf("Enter the element to added in the queue\n");
```

```
                scanf("%d",&val);
                q=insert(q,val);
                break;
            case 2:
                q=delete_element(q);
                break;
            case 3:
                val=peek(q);
                if(val!=-1)
                    printf("The value at the front of queue is %d\n",val);
                break;
            case 4:
                q=display(q);
                break;
        }
    }while(ch>=1&&ch<=4);
    getch();
    return 0;
}
void create_queue(struct queue *q)
{
    q->rear=NULL;
    q->front=NULL;
}
struct queue *insert(struct queue *q,int val)
{
    struct node *ptr;
    ptr=(struct node *)malloc(sizeof(struct node));
    ptr->data=val;
    if(q->front==NULL)
    {
        q->front=ptr;
        q->rear=ptr;
        q->front->next=NULL;
        q->rear->next=NULL;
```

```
	}
	else
	{
		q->rear->next=ptr;
		q->rear=ptr;
		q->rear->next=NULL;
	}
	return q;
}
struct queue *display(struct queue *q)
{
	struct node *ptr;
	ptr=q->front;
	if(ptr==NULL)
		printf("Queue is empty\n");
	else
	{
		printf("\n");
		while(ptr!=q->rear)
		{
			printf("%d\t",ptr->data);
			ptr=ptr->next;
		}
		printf("%d\t",ptr->data);
	}
	return q;
}
struct queue *delete_element(struct queue *q)
{
	struct node *ptr;
	ptr=q->front;
	if(q->front==NULL)
		printf("Underflow\n");
	else
	{
```

```
                q->front=q->front->next;
                printf("The value being deleted is %d\n",ptr->data);
                free(ptr);
        }
        return q;
}
int peek(struct queue *q)
{
        if(q->front==NULL)
        {
                printf("Queue is empty\n");
                return -1;
        }
        else
                return q->front->data;
}
```

5. Quick Sort

```
#include<stdio.h>
#include<conio.h>
void quick_sort(int a[],int l,int r);
int split(int a[],int l,int r);
int main()
{
	int a[10],i,j,n,t;
	printf("Enter how many elements\n");
	scanf("%d",&n);
	printf("Enter the elements\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	quick_sort(a,0,n-1);
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
	getch();
	return 0;
}
void quick_sort(int a[],int l,int r)
{
	int i;
	if(r>l)
	{
		i=split(a,l,r);
		quick_sort(a,l,i-1);
		quick_sort(a,i+1,r);
	}
}
int split(int a[],int l,int r)
```

```
{
        int i,p,q,t;
        p=l+1;
        q=r;
        i=a[l];
        while(q>=p)
        {
                while((a[p]<i)&&(q>=p))
                        p++;
                while((a[q]>i)&&(q>=p))
                        q--;
                if(q>p)
                {
                        t=a[p];
                        a[p]=a[q];
                        a[q]=t;
                }
        }
        t=a[l];
        a[l]=a[q];
        a[q]=t;
        return q;
}
```

6. Selection Sort

```
#include<stdio.h>
#include<conio.h>
int main()
{
	int a[10],i,j,n,t;
	printf("Enter how many elements\n");
	scanf("%d",&n);
	printf("Enter the elements\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}
	for(i=0;i<n-1;i++)
	{
		for(j=i+1;j<=n-1;j++)
		{
			if(a[i]>a[j])
			{
				t=a[i];
				a[i]=a[j];
				a[j]=t;
			}
		}
	}
	printf("Array after sorting is\n");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
	getch();
}
```

7. Insertion Sort

```
#include<stdio.h>
#include<conio.h>
void insertion_sort(int a[],int n);
int main()
{
	int a[10],i,n;
	printf("Enter the number of elements\n");
	scanf("%d",&n);
	printf("Enter the elements to be sorted\n");
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	insertion_sort(a,n);
	printf("Sorted array is\n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);
	getch();
	return 0;
}
void insertion_sort(int a[],int n)
{
	int i,j,t;
	for(i=1;i<n;i++)
	{
		t=a[i];
		j=i-1;
		while((t<a[j])&&(j>=0))
		{
			a[j+1]=a[j];
			j--;
		}
		a[j+1]=t;
	}
}
```

8. One Way Merge Sort

```
#include<stdio.h>
#include<conio.h>
void merge_sort(int a[],int l,int h);
void merge(int a[],int l,int m,int h);
int main()
{
	int a[100],i,n;
	printf("Enter no of elements\n");
	scanf("%d",&n);
	printf("Enter the elements\n");
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	merge_sort(a,0,n-1);
	printf("Sorted array\n");
	for(i=0;i<n;i++)
		printf("%d\t",a[i]);
	getch();
	return 0;
}
void merge_sort(int a[],int l,int h)
{
	int m;
	if(l<h)
	{
		m=(l+h)/2;
		merge_sort(a,l,m);
		merge_sort(a,m+1,h);
		merge(a,l,m,h);
	}
}
void merge(int a[],int l,int m,int h)
{
	int i=l,j=m+1,t=l,c[100],k;
```

```
	while((i<=m)&&(j<=h))
	{
		if(a[i]<a[j])
		{
			c[t]=a[i];
			i++;
		}
		else
		{
			c[t]=a[j];
			j++;
		}
		t++;
	}
	if(i>m)
	{
		while(j<=h)
		{
			c[t]=a[j];
			j++;
			t++;
		}
	}
	else
	{
		while(i<=m)
		{
			c[t]=a[i];
			i++;
			t++;
		}
	}
	for(k=l;k<t;k++)
		a[k]=c[k];
}
```

9. Linear and Binary Search

1.Linear Search

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int n,num,i,f=0,a[10];
	clrscr();
	printf("Entre how many elements\n");
	scanf("%d",&n);
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	printf("Enter the elements to serach\n");
	scanf("%d",&num);
	for(i=0;i<n;i++)
	{
		if(a[i]==num)
		{
			f=1;
			break;
		}
	}
	if(f==1)
	{
		printf("Element is found at %d position",i);
	}
	else
	{
		printf("Element is not found");
	}
	getch();
}
```

2.Binary Search

```
#include<stdio.h>
#include<conio.h>
void main()
{
	int i,a[10],n,l=0,h=0,m,num,f=1;
	clrscr();
	printf("Enter how many elements\n");
	scanf("%d",&n);
	printf("Enter sorted elements\n");
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	printf("Enter the element to search\n");
	scanf("%d",&num);
	h=n-1;
	for(m=(l+h)/2;l<=h;m=(l+h)/2)
	{
		if(a[m]==num)
		{
			printf("Element is found at %d\n",m);
			f=0;
			break;
		}
		if(a[m]>num)
			h=m-1;
		else
			l=m+1;
	}
	if(f)
		printf("Not Found\n");
	getch();
}
```