1. Reverse of a String

```
#include<stdio.h>
int main()
{
char str[100];
char rev[100];
char*sptr=str;
char*rptr=rev;
int i=-1;
printf("\n\nEnter a string:");
scanf("%s",str);
while(*sptr)
{
sptr++;
i++;
}
while(i>=0)
{
sptr--;
*rptr=*sptr;
rptr++;
i--;
}
*rptr='\0';
rptr=rev;
while(*rptr)
{
*sptr=*rptr;
sptr++;
rptr++;
}
printf("\n\nReverse of the string is:%s",str);
printf("\n\n\t\t\tcoding is fun !\n\n\n");
return 0;
}
```

2. Linear Search in 2D Array

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

int main()
{
    int n,m,i,j,s,c=0;
printf("AIM : To Search an element in the 2D Array using linear search.\nEnter the size of  Matrix.\n");
```

```
        scanf("%d%d",&n,&m);
        int **a;
        a = malloc(n*sizeof(int*));
        printf("Enter the elements in the Matrix.\n");
        for(i=0;i<n;i++)
        {
            a[i] = malloc(n*sizeof(int));
            for(j=0;j<m;j++)
            {
                printf("[%d,%d] : ",i+1,j+1);
                scanf("%d",&a[i][j]);
            }
        }
        printf("Enter the element to be searched.\n");
        scanf("%d",&s);
        for(i=0;i<n;i++)
        {
            for(j=0;j<m;j++)
            {
                if(s == a[i][j])
                {
                    printf("Element %d at position Number [%d,%d].\n",s,i+1,j+1);
                    c++;
                }
            }
        }
        if(c==0)
        {
            printf("No Such Element is Present\n");
        }
        return 0;
    }
```

3. Recursive Binary Search

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

int main()
{
    int n,m,i,j,s,c=0;
    printf("AIM : To Search an element in the 2D Array using linear search.\nEnter the size of Matrix.\n");
    scanf("%d%d",&n,&m);
    int **a;
    a = malloc(n*sizeof(int*));
    printf("Enter the elements in the Matrix.\n");
```

```
    for(i=0;i<n;i++)
    {
        a[i] = malloc(n*sizeof(int));
        for(j=0;j<m;j++)
        {
            printf("[%d,%d] : ",i+1,j+1);
            scanf("%d",&a[i][j]);
        }
    }
    printf("Enter the element to be searched.\n");
    scanf("%d",&s);
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            if(s == a[i][j])
            {
                printf("Element %d at position Number [%d,%d].\n",s,i+1,j+1);
                c++;
            }
        }
    }
    if(c==0)
    {
        printf("No Such Element is Present\n");
    }
    return 0;
}
```

4. Append Two Arrays

```
    include<stdio.h>
    #include<conio.h>
    void main()
    {
            clrscr();
            int arr1[50], arr2[50], size1, size2, size, i, j, k, merge[100];
            printf("Enter Array 1 Size : ");
            scanf("%d",&size1);
            printf("Enter Array 1 Elements : ");
            for(i=0; i<size1; i++)
            {
                    scanf("%d",&arr1[i]);
            }
            printf("Enter Array 2 Size : ");
            scanf("%d",&size2);
```

```
        printf("Enter Array 2 Elements : ");
        for(i=0; i<size2; i++)
        {
                scanf("%d",&arr2[i]);
        }
        for(i=0; i<size1; i++)
        {
                merge[i]=arr1[i];
        }
        size=size1+size2;
        for(i=0, k=size1; k<size && i<size2; i++, k++)
        {
                merge[k]=arr2[i];
        }
        printf("Now the new array after merging is :\n");
        for(i=0; i<size; i++)
        {
                printf("%d  ",merge[i]);
        }
        getch();
}
```

5. Polynomial Representation

```
#include<stdio.h>
#include<math.h>
struct poly
{
 float coeff;
 int exp;
};


struct poly a[50],b[50];     //declaration of polynomials

int main()
{
 int i;
 int deg1,deg2;
 int k=0,l=0,m=0;


 printf("Enter the highest degree of polynimial:");
 scanf("%d",&deg1);
 for(i=0;i<=deg1;i++)
 {
```

```
  printf("\nEnter the coeff of x^%d :",i);
  scanf("%f",&a[i].coeff);

        a[k++].exp = i;
}

 printf("\ Polynomial is  %.1f",a[0].coeff);
 for(i=1;i<=deg1;i++)
 {
  printf("+ %.1fx^%d",a[i].coeff,a[i].exp);
 }


 return 0;

}
```

6. Polynomial Addition

```
#include<stdio.h>
#include<math.h>
struct poly
{
 float coeff;
 int exp;
};


struct poly a[50],b[50],c[50],d[50];     //declaration of polynomials

int main()
{
 int i;
 int deg1,deg2;                    //stores degrees of the polynomial
 int k=0,l=0,m=0;


 printf("Enter the highest degree of poly1:");
 scanf("%d",&deg1);                  //taking polynomial terms from the user
 for(i=0;i<=deg1;i++)
 {                           //entering values in coefficient of the polynomial terms
  printf("\nEnter the coeff of x^%d :",i);
  scanf("%f",&a[i].coeff);
                              //entering values in exponent of the polynomial terms
        a[k++].exp = i;
 }
```

7. Stack using Array

```
#include<stdio.h>

int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
  //clrscr();
  top=-1;
  printf("\n Enter the size of STACK[MAX=100]:");
  scanf("%d",&n);
  printf("\n\t STACK OPERATIONS USING ARRAY");
  printf("\n\t--------------------------------");
  printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
  do
  {
    printf("\n Enter the Choice:");
    scanf("%d",&choice);
    switch(choice)
    {
      case 1:
      {
        push();
        break;
      }
      case 2:
      {
        pop();
        break;
      }
      case 3:
      {
        display();
        break;
      }
      case 4:
      {
        printf("\n\t EXIT POINT ");
        break;
      }
      default:
      {
        printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
      }

    }
  }
```

```
   while(choice!=4);
   return 0;
}
void push()
{
   if(top>=n-1)
   {
      printf("\n\tSTACK is over flow");

   }
   else
   {
      printf(" Enter a value to be pushed:");
      scanf("%d",&x);
      top++;
      stack[top]=x;
   }
}
void pop()
{
   if(top<=-1)
   {
      printf("\n\t Stack is under flow");
   }
   else
   {
      printf("\n\t The popped elements is %d",stack[top]);
      top--;
   }
}
void display()
{
   if(top>=0)
   {
      printf("\n The elements in STACK \n");
      for(i=top; i>=0; i--)
         printf("\n%d",stack[i]);
      printf("\n Press Next Choice");
   }
   else
   {
      printf("\n The STACK is empty");
   }

}
```

8. Infix to postfix Conversion

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

```
char stack[20];
int top = -1;
void push(char x)
{
   stack[++top] = x;
}

char pop()
{
   if(top == -1)
      return -1;
   else
      return stack[top--];
}

int priority(char x)
{
   if(x == '(')
      return 0;
   if(x == '+' || x == '-')
      return 1;
   if(x == '*' || x == '/')
      return 2;
}

main()
{
   char exp[20];
   char *e, x;
   printf("Enter the expression :: ");
   scanf("%s",exp);
   e = exp;
   while(*e != '\0')
   {
      if(isalnum(*e))
         printf("%c",*e);
```

```
        else if(*e == '(')
            push(*e);
        else if(*e == ')')
        {
            while((x = pop()) != '(')
                printf("%c", x);
        }
        else
        {
            while(priority(stack[top]) >= priority(*e))
                printf("%c",pop());
            push(*e);
        }
        e++;
    }
    while(top != -1)
    {
        printf("%c",pop());
    }
}
```

9. Iterative binary Search

```
#include<stdio.h>

#include<conio.h>

void main()

{

 int n, arr[50], first, mid, last, key, loc, i;

 int flag=0;

 clrscr();

 printf("\nEnter number of elements: ");

 scanf("%d", &n);
```

```
printf("\nEnter the elements: ");
for(i=0; i<n; i++)
 scanf("%d", &arr[i]);
first=0;
last=n;
printf("\nEnter the element to be searched: ");
scanf("%d", &key);

while(first<=last)
{
 mid=(first+last)/2;
 if(arr[mid]==key)
 {
  flag=1;
  printf("\nElement found at location %d", mid+1);
  break;
 }
 else if(key>arr[mid])
  first=mid+1;
 else
  last=mid-1;
}
if(flag==0)
printf("\nElement not found");
```

```
getch();

}
```

10. Merge Sort

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

int main()
{
        int a[10],b[10],c[20],n1,n2,i,j,temp,k=0;
        clrscr();

        printf(" Enter the no. of element for 1st array : ");
        scanf("%d",&n1);

  for(i=0;i<n1;i++,k++)
  {
    printf(" Enter element [%d] : ",i+1);
    scanf("%d",&a[i]);
    c[k]=a[i];
      }

  for(i=0;i<n1;i++)
  {
    for(j=i+1;j<n1;j++)
    {
      if(a[i]>a[j])
      {
        temp=a[i];
        a[i]=a[j];
        a[j]=temp;
      }
    }
  }

  printf("\n After sorting 1st array : ");
  for(i=0;i<n1;i++)
  {
    printf("\n Element [%d] = %d",i+1,a[i]);
  }


  printf("\n\n Enter the no. of element for 2nd array :  ");
  scanf("%d",&n2);

  for(i=0;i<n2;i++,k++)
```

```
    {
      printf(" Enter element [%d] : ",i+1);
      scanf("%d",&b[i]);
      c[k]=b[i];
    }
    for(i=0;i<n2;i++)
    {
      for(j=i+1;j<n2;j++)
      {
        if(b[i]>b[j])
        {
          temp=b[i];
          b[i]=b[j];
          b[j]=temp;
        }
      }
    }

    printf("\n After sorting 2nd array : ");
    for(i=0;i<n2;i++)
    {
      printf("\n Element [%d] = %d",i+1,b[i]);
    }
    for(i=0;i<n1+n2;i++)
    {
      for(j=i+1;j<n1+n2;j++)
      {
        if(c[i]>c[j])
        {
          temp=c[i];
          c[i]=c[j];
          c[j]=temp;
        }
      }
    }

    printf("\n\n\n After combined and sorted both array :- ");
    for(i=0;i<n1+n2;i++)
    {
      printf("\n Element [%d] = %d",i+1,c[i]);
    }
    getch();

    return 0;
  }
```

11. Bubble Sort

```
#include <stdio.h>

void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
}

// A function to implement bubble sort
void bubbleSort(int arr[], int n)
{
  int i, j;
  for (i = 0; i < n-1; i++)

      // Last i elements are already in place
      for (j = 0; j < n-i-1; j++)
         if (arr[j] > arr[j+1])
            swap(&arr[j], &arr[j+1]);
}

/* Function to print an array */
void printArray(int arr[], int size)
{
   int i;
   for (i=0; i < size; i++)
      printf("%d ", arr[i]);

}

// Driver program to test above functions
int main()
{
   int arr[] = {64, 34, 25, 12, 22, 11, 90};
   int n = sizeof(arr)/sizeof(arr[0]);
   bubbleSort(arr, n);
   printf("Sorted array: \n");
   printArray(arr, n);
   return 0;}
```

12. Exchange Sort

```
#include<stdio.h>
void main()
{
int array[5];
```

```
int length=5;
int i,j;
int temp;
for(i=0;i<5;i++)
{
printf("enter number:");
scanf("%d",&array[i]);
}
for(i=0;i<(length-1);i++)
{
for(j=(i+1);j<length;j++)
{
if(array[i]<array[j])
{
temp=array[i];
array[i]=array[j];
array[j]=temp;
}
}
}
for(i=0;i<5;i++)
{printf("%d ",array[i]);
}
}
```

13. Insertion Sort

```
#include <stdio.h>
#include <conio.h>
void main()
{
        int temp,i,j,num,arr[20];
        clrscr();
        printf("Enter the size of array:");
        scanf("%d",&num);

        printf("Enter %d elements in arr:\n",num);

        for (i = 0; i < num; i++)
        {
        scanf("%d",&arr[i]);
        }
        for(i=1;i<num;i++)
        {
        temp=arr[i];
        j=i-1;
        while((temp<arr[j])&&(j>=0))
        {
        arr[j+1]=arr[j];
        j=j-1;
        }
        arr[j+1]=temp;
        }
printf("After sorting elements:");
for(i=0;i<num;i++)
{
printf("%d ",arr[i]);
}
getch();
}
```

14.Selection Sort

```
void swap(int *xp, int *yp)
{
   int temp = *xp;
   *xp = *yp;
   *yp = temp;
```

```
}

void selectionSort(int arr[], int n)
{
   int i, j, min_idx;

   // One by one move boundary of unsorted subarray
   for (i = 0; i < n-1; i++)
   {
      // Find the minimum element in unsorted array
      min_idx = i;
      for (j = i+1; j < n; j++)
        if (arr[j] < arr[min_idx])
          min_idx = j;

      // Swap the found minimum element with the first element
      swap(&arr[min_idx], &arr[i]);
   }
}
/* Function to print an array */
void printArray(int arr[], int size)
{
   int i;
   for (i=0; i < size; i++)
      printf("%d ", arr[i]);
   printf("\n");
}

// Driver program to test above functions
int main()
{
   int arr[] = {64, 25, 12, 22, 11};
   int n = sizeof(arr)/sizeof(arr[0]);
   selectionSort(arr, n);
   printf("Sorted array: \n");
   printArray(arr, n);
   return 0;
}
```

15.Linked List

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

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

```
struct node *head = NULL;
struct node *current = NULL;

//display the list
void printList() {

   struct node *ptr = head;

   printf("\n[head]->");
   //start from the beginning
   while(ptr != NULL) {
      printf("%d->",ptr->data);
      ptr = ptr->next;
   }

   printf(" [null]\n");
}

//insert link at the first location
void insert(int data) {
   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct node));

   //link->key = key;
   link->data = data;

   //point it to old first node
   link->next = head;

   //point first to new first node
   head = link;
}

int main()
{

  int n,i,item;
  printf("How many integers you want to insert into the Linked List:");
  scanf("%d",&n);
  for (i=0;i<n;i++)
  {
     printf("Item[%d]:",i+1);
     scanf("%d",&item);
     insert(item);
  }

   printf("\n\n Display the Linked List");
```

```
  printf("\n-----------------------\n ");
  printList();
  return 0;
}
```