1. 2D matrix multiplication.

```
#include<stdio.h>

int main( ) {
 int m,n,p,q,c,d,k,sum=0;
 int first[10][10], second[10][10],multiply[10][10];

 printf("Enter number of rows and columns of First matrix\n");
 scanf("%d%d",&m,&n);

 printf("Enter the elements of First matrix: \n");
 for(c=0;c<m;c++)
    for(d=0;d<n;d++)
    scanf("%d",&first[c][d]);

 printf("Enter number of rows and columns of Second matrix\n");
 scanf("%d%d",&p,&q);

 if(n!=p)
    printf("The matrix can't be multiplied with each other.\n");

 else{
    printf("Enter the elements of Second matrix: \n");

 for(c=0;c<p;c++)
    for(d=0;d<q;d++)
    scanf("%d",&second[c][d]);

 for(c=0;c<m;c++){
    for(d=0;d<q;d++){
        for(k=0;k<p;k++){
                sum=sum+(first[c][k]*second[k][d]);
                multiply[c][d]=sum;
                        }
                  sum=0;
                    }
                }

    printf("Product of the matrix: \n");

    for(c=0;c<m;c++){
        for(d=0;d<q;d++){
        printf("%d\t",multiply[c][d]);
        printf("\n");
        }
    }
 }
    return 0;
    }
```

Sample Output:

```
Enter number of rows and columns of First matrix
2 2
Enter the elements of First matrix:
10 20 30 40
Enter number of rows and columns of Second matrix
2 2
Enter the elements of Second matrix:
10 20 30 40
Product of the matrix:
700
1000
1500
2200

Process returned 0 (0x0)   execution time : 90.084 s
Press any key to continue.
```

2. Sorting Number (Descending order).

```
#include<stdio.h>

int main()
    {
        int a[100],x,n,i,j;
        printf("Enter the Number of values: ");
        scanf("%d",&n);

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

        for(i=0;i<n;i++){
            for(j=i+1;j<n;j++)
            {
                if(a[i]<a[j]){
                    x=a[i];
                    a[i]=a[j];
                    a[j]=x;
                }
            }
        }
        printf("\n Here is your values shorted in descending order: \n");

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

    return 0;

    }
```

Sample Output:

```
Enter the Number of values: 5
10 20 30 40 50

 Here is your values shorted in descending order:
50
40
30
20
10

Process returned 0 (0x0)   execution time : 8.485 s
Press any key to continue.
```

3. Find the missing number.

```
#include <stdio.h>
    int main()
    {
        int n, i, j, c, t, b;

    printf("Enter size of array : ");
    scanf("%d", &n);
        int array[n - 1];
        printf("Enter elements into array : \n");
        for (i = 0; i < n - 1; i++)
            scanf("%d", &array[i]);

    b = array[0];
        for (i = 1; i < n - 1; i++)
            b = b ^ array[i];
        for (i = 2, c = 1; i <= n; i++)
            c = c ^ i;
        c = c ^ b;

    printf("Missing element is : %d \n", c);
    return 0;
    }
```

Sample Output:

```
Enter size of array : 5
Enter elements into array :
1 2 3 5 6
Missing element is : 4

Process returned 0 (0x0)   execution time : 12.640 s
Press any key to continue.
```

4. Substring Check.

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

int main()
{
    char string[100], substring[100];
    int stringlen = 0, substringlen = 0, i, j, flag;

    printf("Enter a string: ");
    gets(string);

    printf("Enter substring to search: ");
    gets(substring);

    stringlen=strlen(string);
    substringlen=strlen(substring);

    for (i = 0; i <= stringlen - substringlen; i++)
    {
        for (j = i; j < i + substringlen; j++)
        {
            flag = 1;
            if (string[j] != substring[j - i])
            {
                flag = 0;
                break;
            }
        }
        if (flag == 1)
            break;
    }

    printf("\n");

    if (flag == 1)
        printf("String Found.\n");
    else
        printf("String Not Found.\n");

    return 0;
}
```

Sample Output:

1.

```
Enter a string: bangla
Enter substring to search: desh

String Not Found.

Process returned 0 (0x0)   execution time : 5.261 s
Press any key to continue.
```

2.

```
Enter a string: bangla
Enter substring to search: gla

String Found.

Process returned 0 (0x0)   execution time : 5.506 s
Press any key to continue.
```

5. Insert a pattern into a text.

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

int insertString(char a[],char b[],int pos);

int main()
{
  char text[100],pattern[100];
  int position;

  printf("Enter the Text: ");
  scanf("%s",&text);
  printf("Enter the Pattern: ");
  scanf("%s",&pattern);
  printf("Enter the Position: ");
  scanf("%d",&position);

  int result=insertString(text,pattern,position);
  if(result!=-1)
    printf("String  is : %s \n",text);
  else
    printf("Not possible\n");
  return 0;

}

int insertString(char a[],char b[],int pos)
{
  int i=0,j=0;

  int lengthA=strlen(a);
  int lengthB=strlen(b);

  if(pos>lengthA)
    return -1;

  for(i=lengthA;i>=pos;i--)
      a[i+lengthB]=a[i];

  for ( i = 0; i < lengthB; ++i )
      a[i + pos] = b[i];

  return 1;
}
```

Sample Output:

```
Enter the Text: Bangla
Enter the Pattern: desh
Enter the Position: 6
String  is : Bangladesh

Process returned 0 (0x0)   execution time : 10.108 s
Press any key to continue.
```

6. Find a substring of a text.

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

int main()
{
     char string[100],substring[100];
     int n,position,l,i,j;

     puts("Enter a string :");
     gets(string);
     printf("Enter the position from where you want to start: ");
     scanf("%d",&position);
     printf("Enter the number of characters: ");
     scanf("%d",&n);

    l=strlen(string);

    if(position>0&&n<=l){
    for(j=0,i=position-1;i<=(position+n)-1;i++,j++){
        substring[j]=string[i];
    }
    substring[i]='\0';

    printf("\nSubstring: %s\n",substring);
    }

    else{
        printf("\nNot Possible.\n");
    }
    return 0;
}
```

Sample Output:

```
Enter a string :
DataStructure
Enter the position from where you want to start: 5
Enter the number of characters: 9

Substring: Structure

Process returned 0 (0x0)   execution time : 10.833 s
Press any key to continue.
```

7. Delete a pattern from a text.

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

int main()
{
    int i, j = 0, k = 0,n = 0;
    int flag = 0;
    char text[100], newstring[100], pattern[100];

    printf("\nEnter the text: ");
    gets(text);

    printf("\nEnter the pattern you want to be Removed: ");
    gets(pattern);

    for(i = 0 ; text[i] != '\0' ; i++)
    {   k = i;

        while(text[i] == pattern[j])
        {
            i++,j++;
            if(j == strlen(pattern))
            {
                flag = 1;
                break;
            }
        }
    j = 0;

    if(flag == 0)
        i = k;
    else
        flag = 0;

    newstring[n++] = text[i];
    }

    newstring[n] = '\0';
    printf("\n\nAfter removing Pattern from text: %s\n",newstring);

    return 0;
}
```

Sample Output:

```
Enter a string :
Bangladesh
Enter the position from where you want to delete:7
Enter the number of characters to be deleted :4

Bangla

Process returned 0 (0x0)   execution time : 12.119 s
Press any key to continue.
```

8. Replace a pattern from a text.

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

void replaceSubstring(char [],char[],char[]);

    int main()
    {
        char string[100],sub[100],new[100];

        printf("\nEnter a string: ");
        gets(string);
        printf("\nEnter the substring: ");
        gets(sub);
        printf("\nEnter the new substring: ");
        gets(new);
        replaceSubstring(string,sub,new);

        printf("\nThe string after replacing: \n\n %s \n\n",string);

    return 0;
    }

void replaceSubstring(char string[],char sub[],char new[])
    {
    int stringLen,subLen,newLen;
    int i=0,j,k;
    int flag=0,start,end;

    stringLen=strlen(string);
    subLen=strlen(sub);
    newLen=strlen(new);

    for(i=0;i<stringLen;i++)
    {
          flag=0;
          start=i;

    for(j=0;string[i]==sub[j];j++,i++) /* Checks for the substring */
          if(j==subLen-1)
    flag=1; /* flag sets when substring is identified */
          end=i;
    if(flag==0)
          i-=j;
    else
         {
          for(j=start;j<end;j++) /* Delete the substring */
               {
           for(k=start;k<stringLen;k++)
          string[k]=string[k+1];
          stringLen--;
          i--;
               }
```

```
            for(j=start;j<start+newLen;j++)    /* Inserting new substring
   */
            {
            for(k=stringLen;k>=j;k--)
           string[k+1]=string[k];
           string[j]=new[j-start];
            stringLen++;
           i++;
          }
                 }
            }
      }
```

Sample Output:

```
Enter a string: Bangladesh

Enter the substring: desh

Enter the new substring: city

The string after replacing:

 Banglacity


Process returned 0 (0x0)   execution time : 112.171 s
Press any key to continue.
```

9. 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]) /* For decreasing order use < */
      {
        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;
}
```

Sample Output:

```
Enter number of elements
5
Enter 5 integers
50 40 30 20 10
Sorted list in ascending order:
10
20
30
40
50

Process returned 0 (0x0)   execution time : 15.741 s
Press any key to continue.
```

10. 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;
}
```

Sample Output:

```
Enter number of elements
5
Enter 5 integers
50 40 30 20 10
Sorted list in ascending order:
10
20
30
40
50

Process returned 0 (0x0)   execution time : 10.663 s
Press any key to continue.
```

11. Linear Search.

```
#include <stdio.h>

int main()
{
  int array[100], search, i, n;

  printf("Enter number of elements in array: \n");
  scanf("%d", &n);

  printf("Enter %d integers: \n", n);

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

  printf("Enter a number to search\n");
  scanf("%d", &search);

  for (i = 0; i < n; i++)
  {
    if (array[i] == search)
    {
      printf("%d is present at location %d.\n", search, i+1);
      break;

    }
  }

  if (i == n)
    printf("%d isn't present in the array.\n", search);

  return 0;
}
```

Sample Output:

```
Enter number of elements in array:
5
Enter 5 integers:
88 55 99 11 22
Enter a number to search
99
99 is present at location 3.

Process returned 0 (0x0)   execution time : 22.649 s
Press any key to continue.
```

12. Binary Search.

```
#include <stdio.h>

int main()
{
   int i, first, last, middle, n, search, array[100];

   printf("Enter number of elements:\n");
   scanf("%d",&n);

   printf("Enter %d integers: \n", n);

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

   printf("Enter value to find: \n");
   scanf("%d", &search);

   first = 0;
   last = n - 1;
   middle = (first+last)/2;

   while (first <= last) {
      if (array[middle] < search)
         first = middle + 1;
      else if (array[middle] == search) {
         printf("%d found at location %d.\n", search, middle+1);
         break;
      }
      else
         last = middle - 1;

      middle = (first + last)/2;
   }
   if (first > last)
      printf("Not found! %d isn't present in the list.\n", search);

   return 0;
}
```

Sample Output:

```
Enter number of elements:
5
Enter 5 integers:
10 20 30 40 50
Enter value to find:
30
30 found at location 3.

Process returned 0 (0x0)   execution time : 49.150 s
Press any key to continue.
```

13. Linked list implementation.

I. Create a simple linked list.

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

   int main()
   {
      struct node a,b,c;
      struct node* i;
      struct node addfirst,addlast;
      int item;

      a.value=100;
      b.value=200;
      c.value=300;

      a.next=&b;
      b.next=&c;
      c.next=NULL;

      for(i=&a;i!=NULL;i=i->next)
      printf("%d ",i->value);

      return 0;
   }
```

II. Insertion of a node into Linked list.

```
#include<stdio.h>

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

    int main()
    {
    struct node a,b,c;
    struct node* i;
    struct node addfirst,addlast;
    int item;

    a.value=100;
    b.value=200;
    c.value=300;
```

```
a.next=&b;
b.next=&c;
c.next=NULL;

for(i=&a;i!=NULL;i=i->next)
printf("%d ",i->value);

printf("\nAdd first\n");
scanf("%d",&item);

addfirst.value=item;
addfirst.next=&a;

for(i=&addfirst;i!=NULL;i=i->next)
printf("%d ",i->value);

printf("\nAdd last\n");
scanf("%d",&item);

c.next=&addlast;
addlast.value=item;
addlast.next=NULL;

for(i=&addfirst;i!=NULL;i=i->next)
printf("%d ",i->value);


return 0;
}
```

Sample Output:

```
100 200 300
Add first
50
50 100 200 300
Add last
400
50 100 200 300 400
Process returned 0 (0x0)   execution time : 7.065 s
Press any key to continue.
```

14. Stack implementation using array.

Operations:

I. Push
II. Pop
III. Display
IV. Exit

```
#include<stdio.h>

int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{

    top=-1;
    printf("\n Enter the size of STACK:");
    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");
    }

}
```

Sample Output:

```
Enter the size of STACK:5

        STACK OPERATIONS USING ARRAY
       --------------------------------
        1.PUSH
        2.POP
        3.DISPLAY
        4.EXIT
Enter the Choice:3

The STACK is empty
Enter the Choice:1
Enter a value to be pushed:50

Enter the Choice:1
Enter a value to be pushed:60

Enter the Choice:2

        The popped elements is 60
Enter the Choice:3

The elements in STACK

50
Press Next Choice
Enter the Choice:1
Enter a value to be pushed:70

Enter the Choice:3

The elements in STACK

70
50
Press Next Choice
Enter the Choice:4

        EXIT POINT
Process returned 0 (0x0)   execution time : 112.089 s
Press any key to continue.
```

15. Queue implementation

Operations:

I. Insert( )
II. Delete( )
III. Display the front element
IV. Is empty( )
V. Is full ( )
VI. Exit.

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

#define max 5
int q[max],front=0,rear=-1;

void main( )
{
    int ch;
    void insert( );
    void delet( );
    void display( );
    void isempty( );
    void isfull( );

    printf("\nQueue implementation.\n");

    printf("1.Insert\n2.Delete\n3.Display the front element\n4.Is
empty? \n5.Is Full?\n6.exit\n");

    while(1)
    {
        printf("Enter your choice:");
        scanf("%d",&ch);

        switch(ch)
        {
        case 1: insert( );
            break;
        case 2: delet( );
            break;
        case 3:display( );
            break;
        case 4:
            isempty( );
            break;
        case 5:
            isfull( );
            break;
        case 6:
            exit(0);

        default:printf("Invalid option\n");

        }
```

```
    }

return 0;
}



void insert()
{
    int x;
    if((front==0&&rear==max-1)||(front>0&&rear==front-1))
        printf("Queue is overflow\n");

    else
    {
        printf("Enter element to be insert:");
        scanf("%d",&x);

        if(rear==max-1&&front>0)
        {
            rear=0;
            q[rear]=x;
        }
        else
        {
            if((front==0&&rear==-1)||(rear!=front-1))
                q[++rear]=x;
        }
    }
}

void  delet()
{
    int a;
    if((front==0)&&(rear==-1))
    {
        printf("Queue is underflow\n");

        exit(0);
    }
    if(front==rear)
    {
        a=q[front];
        rear=-1;
        front=0;
    }
    else
        if(front==max-1)
        {
            a=q[front];
            front=0;
        }
        else a=q[front++];
        printf("Deleted element is:%d\n",a);
}
```

```
void display()
{
    int i,j;
    if(front==0&&rear==-1)
    {
        printf("Queue is underflow\n");

        exit(0);
    }
    if(front>rear)
    {
        for(i=0;i<=rear;i++)
            printf("\t%d",q[i]);
        for(j=front;j<=max-1;j++)
            printf("\t%d",q[j]);
        //printf("\nrear is at %d\n",q[rear]);
        printf("\nfront is at %d\n",q[front]);
    }
    else
    {
        for(i=front;i<=rear;i++)
        {
            printf("\t%d",q[i]);
        }
        //printf("\nrear is at %d\n",q[rear]);
        printf("\nfront is at %d\n",q[front]);
    }
    printf("\n");
}

 void isfull(){

    if((front==0&&rear==max-1)||(front>0&&rear==front-1))
        printf("Queue is Full\n\n");

    else
        printf("Queue is Not Full.\n\n");
 }

 void isempty(){

     if(front==0&&rear==-1)
    {
        printf("Queue is Empty.\n\n");

    }
    else
        printf("Queue is Not Empty.\n\n");
 }
```

Sample Output:

```
Queue implementation.
1.Insert
2.Delete
3.Display the front element
4.Is empty?
5.Is Full?
6.exit
Enter your choice:4
Queue is Empty.

Enter your choice:1
Enter element to be insert:10
Enter your choice:1
Enter element to be insert:20
Enter your choice:3
        10      20
front is at 10

Enter your choice:4
Queue is Not Empty.

Enter your choice:6

Process returned 0 (0x0)   execution time : 67.730 s
Press any key to continue.
```

16. Graph implementation

```
#include<stdio.h>
int main()
{
    int i,j,sum=0,n,m[10][10];

    printf("Enter the number of nodes: ");
    scanf("%d",&n);

    printf("Number of elements :\n");
    for(i=0;i<n;i++)
    for(j=0;j<n;j++)
    scanf("%d",&m[i][j]);

    for(i=0;i<n;i++){
            sum=0;
        for(j=0;j<n;j++){
        sum=sum+m[j][i];
    }
    printf("Out-degree %d: %d\n",i+1,sum);

    }
    printf("\n");

    for(i=0;i<n;i++){
        sum=0;
        for(j=0;j<n;j++){
            sum=sum+m[i][j];
        }
        printf("In-degree %d: %d\n",i+1,sum);
    }
    return 0;
}
```

Sample Output:

```
Enter the number of nodes: 4
Number of elements :
1 1 0 0
1 1 0 0
0 0 1 1
0 0 1 1
Out-degree 1: 2
Out-degree 2: 2
Out-degree 3: 2
Out-degree 4: 2

In-degree 1: 2
In-degree 2: 2
In-degree 3: 2
In-degree 4: 2

Process returned 0 (0x0)   execution time : 28.388 s
Press any key to continue.
```