A1) - MERGE SORT (CHARACTERS)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//Ascending sort
char a[100];
void merge(int low,int mid,int high){
    int b[100];
    int h,i,j,k;
    h = low;
    i = low;
    j = mid+1;
    while((h <= mid) && (j <= high))
    {
        if(a[h]<=a[j])
        {
            b[i] = a[h];
            h = h+1;
        }
        else
        {
            b[i] = a[j];
            j = j + 1;
        }
        i = i+1;
    }
    if(h > mid){
        for(k = j; k <= high; k++)
        {
            b[i] = a[k];
            i = i + 1;
        }
```

```
    }
    else{
        for(k = h; k <= mid; k++){
            b[i] = a[k];
            i = i+1;
        }
    }
    for( k = low; k <= high; k++)
    {
        a[k] = b[k];
    }
}
void mergeSort(int low,int high,int n)
{
    int mid,i;
    if(low<high)
    {
        mid = (low + high)/2;
        mergeSort(low,mid,n);
        mergeSort(mid+1,high,n);
        for(i = 0; i < n; i++){
            printf(" %c",a[i]);
        }
        printf("\n");
        merge(low,mid,high);
    }}
void display(int n){
    int i;
    for(i = 0; i < n; i++)
    {
        printf(" %c",a[i]);
    }}
```

```
int main(){
    int n,i,lo,hi,pos=0;
    char x;
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter Element : ");
    for(i = 0; i < n; i++){
        scanf(" %c",&a[i]);
    }
    lo = 0;
    hi = n-1;
    printf("\nMerge Sort Iterations : \n");
    mergeSort(lo,hi,n);
    printf("\nAfter Sort: ");
    display(n);
    return 0;
}
```

Output :

```
Enter number of elements
11
Enter Element : I T B M Z F S U G H Q
Merge Sort Iterations :
I T B M Z F S U G H Q
 I T B M Z F S U G H Q
 B I T M Z F S U G H Q
 B I T M Z F S U G H Q
 B I T F M Z S U G H Q
 B F I M T Z S U G H Q
 B F I M T Z S U G H Q
 B F I M T Z G S U H Q
 B F I M T Z G S U H Q
 B F I M T Z G H Q S U

After Sort:  B F G H I M Q S T U Z
```

A2) - MERGE SORT (INTEGERS)

```
#include <stdio.h>
#include <stdlib.h>
//Descending sort of numbers
int a[100];
void merge(int low,int mid,int high){
    int b[100];
    int h,i,j,k;
    h = low;
    i = low;
    j = mid+1;
    while((h <= mid) && (j <= high)){
        if(a[h]>=a[j]){
            b[i] = a[h];
            h = h+1;
        }
        else{
            b[i] = a[j];
            j = j + 1;
        }
        i = i+1;
    }
    if(h > mid){
        for(k = j; k <= high; k++)
        {
            b[i] = a[k];
            i = i + 1;
        }
    }
    else{
        for(k = h; k <= mid; k++){
            b[i] = a[k];
```

```
            i = i+1;
        }}
    for( k = low; k <= high; k++)
    {
        a[k] = b[k];
    }
}
void mergeSort(int low,int high,int n){
    int mid,i;
    if(low<high){
        mid = (low + high)/2;
        mergeSort(low,mid,n);
        mergeSort(mid+1,high,n);
        for(i = 0; i < n; i++)
        {
            printf("%d ",a[i]);
        }
        printf("\n");
        merge(low,mid,high);
    }}
void display(int n)
{
    int i;
    for(i = 0; i < n; i++)
    {
        printf("%d ",a[i]);
    }}
int main(){
    int n,i,lo,hi;
    printf("Enter number of elements\n");
    scanf("%d",&n);
    for(i = 0; i < n; i++)
```

```
    {
        printf("Enter Element : ");
        scanf("%d",&a[i]);
    }
    lo = 0;
    hi = n-1;
    printf("\nMerge Sort Iterations (Descendig): \n");
    mergeSort(lo,hi,n);
    printf("\nAfter Sort: ");
    display(n);
    return 0;
}
```

Output:

```
Enter number of elements
11
Enter Elements : 81 43 61 21 -8 96 55 77 -18 52 17
Merge Sort Iterations (Descendig):
81 43 61 21 -8 96 55 77 -18 52 17
81 43 61 21 -8 96 55 77 -18 52 17
81 61 43 21 -8 96 55 77 -18 52 17
81 61 43 21 -8 96 55 77 -18 52 17
81 61 43 96 21 -8 55 77 -18 52 17
96 81 61 43 21 -8 55 77 -18 52 17
96 81 61 43 21 -8 77 55 -18 52 17
96 81 61 43 21 -8 77 55 -18 52 17
96 81 61 43 21 -8 77 55 -18 52 17
96 81 61 43 21 -8 77 55 52 17 -18

After Sort: 96 81 77 61 55 52 43 21 17 -8 -18
```

A3) BINARY SEARCH

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
//binary search
char a[100];
int binarySearch(char a[],int low, int high,char x,int n){
    int mid,i;
    if(low==high){
        if(a[low] == x){
            return low;
        }
        return -1;}
    else{
        mid = (low+high)/2;
        if(a[mid] == x){
            for(i = 0; i < n; i++){
            printf(" %c",a[i]);
            }
            return mid;}
        else if(x > a[mid]){
            for(i = 0; i < n; i++){
            printf(" %c",a[i]);}
            printf("   mid = %d\n",mid);
            return binarySearch(a,mid+1,high,x,n);
        }
        else{
            for(i = low; i < n; i++){
            printf(" %c",a[i]);
            }
            printf("   mid = %d\n",mid);
            return binarySearch(a,low,mid-1,x,n);}}}
int main(){
    int n,i,lo,hi,pos=0;
```

```
    char x;
    printf("Enter number of elements\n");
    scanf("%d",&n);
    printf("Enter Element : ");
    for(i = 0; i < n; i++){
        scanf(" %c",&a[i]);
    }
    lo = 0;
    hi = n-1;
    printf("\n\nEnter element to Search : ");
    scanf(" %c",&x);
    pos = binarySearch(a,lo,hi,x,n);
    if(pos == -1){
        printf("mid = %d",pos);
        printf("\nElement %c not found",x);}
    else{
        printf("   mid = %d",pos);
        printf("\nElement is at position %d",pos);}
    printf("\n");
    return 0;
}
```

Output :

```
Enter number of elements
11
Enter Element : B F G H I M Q S T U Z
Enter element to Search : E
B F G H I M Q S T U Z   mid = 5
 B F G H I M Q S T U Z   mid = 2
 B F G H I M Q S T U Z   mid = 0
mid = -1
Element E not found
```

```
Enter number of elements
11
Enter Element : B F G H I M Q S T U Z
Enter element to Search : Q
B F G H I M Q S T U Z   mid = 5
 Q S T U Z   mid = 8
 B F G H I M Q S T U Z   mid = 6
Element is at position 6
```

B1) MINMAX

```
#include <stdio.h>
#define MAX 9
int a[MAX];
void minmax(int i, int j, int *max, int *min,int n){
    int mid, max1, min1,z;
    if (i == j){
        *max = a[i];
        *min = a[i];
        return;
    }
    else if (i == j - 1){
        if (a[i] > a[j])
        {
            *max = a[i];
            *min = a[j];
        }
        else
        {
            *max = a[j];
            *min = a[i];
        }
        return;
    }
    else{
        mid = (i + j) / 2;
        minmax(i, mid, max, min,n);
        minmax(mid + 1, j, &max1, &min1,n);
        if (max1 > *max)
            *max = max1;
        if (min1 < *min)
            *min = min1;
```

```
    }
}
int main(){
    int i,n, min = 0, max = 0,size;
    printf("Enter no. of elements :\n");
    scanf("%d",&n);
    printf("Enter Elements : ");
    for (i = 0; i <n; i++)
    {
        scanf("%d", &a[i]);
    }
    size = n-1;
    minmax(0, size, &max, &min,n);
    printf("Max is %d\n", max);
    printf("Min is %d\n", min);
}
```

Output :

```
Enter no. of elements :
9
Enter Elements : 43 -12 11 58 -5 29 65 -17 37
Max is 65
Min is -17
```

B2) QUICK SORT

```
#include <stdio.h>
void sw(int a[], int x, int b);
int quicksort(int a[], int p, int q,int size);
int partition(int a[], int p, int q);
int main()
{
    int i,size;
    printf("Enter number of elements : ");
    scanf("%d", &size);
    int a[size];
    printf("Enter elements : ");
    for (i = 0; i < size; i++)
    {
        scanf("%d", &a[i]);
    }

    printf("\nQuick Sort Iterations\n");
    quicksort(a, 0, size - 1,size);
    printf("Sorted array \n");
    for (i = 0; i < size; i++)
        printf("%d, ", a[i]);
    printf("\n");
}

void sw(int a[], int x, int b)
{
    int t = a[x];
    a[x] = a[b], a[b] = t;
}

int quicksort(int a[], int p, int q,int size)
{
    int pivot,i;
```

```
    if (p < q)
    {
        pivot = partition(a, p, q);
        for (i = 0; i < size; i++)
            printf("%d, ", a[i]);
        printf("\n");
        quicksort(a, p, pivot - 1,size);
        quicksort(a, pivot + 1, q,size);
    }
}
int partition(int a[], int p, int q)
{
    int i,j,pivot;
    pivot = a[q], i = p - 1, j;
    for (j = p; j < q; j++){
        if (a[j] < pivot)
        {
            i++;
            sw(a, i, j);
        }}
    sw(a, i + 1, q);
    return i + 1;
}
```

Output :

```
Enter number of elements : 9
Enter elements : 43 -12 11 58 -5 29 65 -17 37
Quick Sort Iterations
-12, 11, -5, 29, -17, 37, 65, 43, 58,
-17, 11, -5, 29, -12, 37, 65, 43, 58,
-17, -12, -5, 29, 11, 37, 65, 43, 58,
-17, -12, -5, 11, 29, 37, 65, 43, 58,
-17, -12, -5, 11, 29, 37, 43, 58, 65,
Sorted array
-17, -12, -5, 11, 29, 37, 43, 58, 65,
```

C1) Kth Smallest Element (int)

```
#include<stdio.h>
#include<stdlib.h>
#include<limits.h>
int a[100];
void sw(int a[], int x, int b){
    int t = a[x];
    a[x] = a[b];
    a[b] = t;
}
int partition(int a[], int p, int q){
    int i,j,pivot;
    pivot = a[q], i = p - 1, j;
    for (j = p; j < q; j++){
        if (a[j] < pivot){
            i++;
            sw(a, i, j);
        }}
    sw(a, i + 1, q);
    return i + 1;
}
int kthsmall(int a[], int n, int k){
    int low,up,j,i;
    low = 1;
    up = n+1;
    a[up] = INT_MAX;
    while(low<=up){
        j = partition(a,low,up);
        if(k == j){
            for (i = 1; i <= n; i++)
            printf("%d, ", a[i]);
            printf("\n");
            return a[j];
        }
```

```
        else if(k > j){
            low = j + 1;}
        else{
            up = j-1;}
        for (i = 1; i <= n; i++)
        printf("%d, ", a[i]);
        printf("\n");
    }}
int main()
{
    int i,size,k,data;
    printf("Enter number of elements : ");
    scanf("%d", &size);
    printf("Enter elements : ");
    for (i = 1; i <= size; i++){
        scanf("%d", &a[i]);
    }
    printf("Which kth smallest to find ?\n");
    scanf("%d", &k);
    printf("\nIterations : \n");
    data = kthsmall(a,size,k);
    printf("The %dth smallest element is %d",k,data);
    return 0;
}
```

Output :

```
Enter number of elements : 11
Enter elements : 33 92 87 43 23 11 79 54 28 69 5
Which kth smallest to find ?
8

Iterations :
33, 92, 87, 43, 23, 11, 79, 54, 28, 69, 5,
5, 92, 87, 43, 23, 11, 79, 54, 28, 69, 33,
5, 23, 11, 28, 33, 87, 79, 54, 43, 69, 92,
5, 23, 11, 28, 33, 87, 79, 54, 43, 69, 92,
5, 23, 11, 28, 33, 54, 43, 69, 79, 87, 92,
The 8th smallest element is 69
```

C2) Kth Smallest Element (char)

```
#include<stdio.h>
#include<stdlib.h>
#include<limits.h>
char a[100];
void sw(char a[], int x, int b){
    char t = a[x];
    a[x] = a[b];
    a[b] = t;
}
int partition(char a[], int p, int q){
    int i,j,pivot;
    pivot = a[q], i = p - 1, j;
    for (j = p; j < q; j++){
        if (a[j] < pivot){
            i++;
            sw(a, i, j);
        }}
    sw(a, i + 1, q);
    return i + 1;   }
char kthsmall(char a[], int n, int k){
    int low,up,j,i;
    low = 1;
    up = n+1;
    a[up] = CHAR_MAX;
    while(low<=up){
        j = partition(a,low,up);
        if(k == j){
            for (i = 1; i <= n; i++)
                printf("%c ", a[i]);
            printf("\n");
            return a[j];  }
        else if(k > j){
            low = j + 1;  }
```

```
        else{
            up = j-1;     }
        for (i = 1; i <= n; i++)
            printf("%c ", a[i]);
        printf("\n");
    }}
int main(){
    int i,size, k;
    char data;
    printf("Enter number of elements : ");
    scanf("%d", &size);
    printf("Enter elements : ");
    for (i = 1; i <= size; i++){
        scanf(" %c", &a[i]);
    }
    printf("Which kth smallest to find ?\n");
    scanf("%d", &k);
    printf("\nIterations : \n");
    data = kthsmall(a,size,k);
    printf("The %dth smallest element is %c",k,data);
    return 0;
}
```

Output:

```
Enter number of elements : 11
Enter elements : K Y W N I G U Q J R E
Which kth smallest to find ?
7

Iterations :
K Y W N I G U Q J R E
E Y W N I G U Q J R K
E I G J K W U Q N R Y
E I G J K W U Q N R Y
E I G J K Q N R U W Y
E I G J K N Q R U W Y
E I G J K N Q R U W Y
The 7th smallest element is Q
```

D1) Strassen’s Matrix Multiplication (4x4)

```
#include <stdio.h>
#include <stdlib.h>
int a[4][4];
int b[4][4];
int c[4][4];
int temp[2][2],temp2[2][2];
int p1[2][2], q1[2][2],r1[2][2],s1[2][2],t1[2][2],u1[2][2],v1[2][2];
void display(int z[2][2]);
void multiple2x2(int temp[2][2],int temp2[2][2],int z[2][2]);
int main(){
    int i, j;
    //Init Matrix
    printf("A Matrix\n");
    for(i = 0; i < 4; i++){
        for(j = 0; j < 4; j++){
            scanf("%d",&a[i][j]);
        }}
    printf("\nB Matrix\n");
    for(i = 0; i < 4; i++){
        for(j = 0; j < 4; j++){
            scanf("%d",&b[i][j]);
        }}
    //P
    for ( i = 0; i < 2; ++i) {
      for ( j = 0; j < 2; ++j) {
            temp[i][j] = 0;
            temp2[i][j] = 0;
            temp[i][j] += ((a[i][j] )+ (a[i+2][j+2]));
            temp2[i][j] += ((b[i][j] )+ (b[i+2][j+2]));
         }}
    multiple2x2(temp,temp2,p1);
    printf("P Matrix\n");
    display(p1);
    //Q
    for ( i = 0; i < 2; ++i) {
```

```
  for ( j = 0; j < 2; ++j) {
        temp[i][j] = 0;
        temp2[i][j] = 0;
        temp[i][j] += ((a[i+2][j] )+ (a[i+2][j+2]));
        temp2[i][j] += (b[i][j]);
     }}
multiple2x2(temp,temp2,q1);
printf("Q Matrix\n");
display(q1);
//R
for ( i = 0; i < 2; ++i) {
  for ( j = 0; j < 2; ++j) {
        temp[i][j] = 0;
        temp2[i][j] = 0;
        temp[i][j] += (a[i][j]);
        temp2[i][j] += (b[i][j+2] - b[i+2][j+2]);
     }}
multiple2x2(temp,temp2,r1);
printf("R Matrix\n");
display(r1);
//S
for ( i = 0; i < 2; ++i) {
  for ( j = 0; j < 2; ++j) {
        temp[i][j] = 0;
        temp2[i][j] = 0;
        temp[i][j] += (a[i+2][j+2]);
        temp2[i][j] += (b[i+2][j] - b[i][j]);
     }}
multiple2x2(temp,temp2,s1);
printf("S Matrix\n");
display(s1);
//T
for ( i = 0; i < 2; ++i) {
  for ( j = 0; j < 2; ++j) {
        temp[i][j] = 0;
        temp2[i][j] = 0;
```

```
         temp[i][j] += ((a[i][j]) + (a[i][j+2]));
         temp2[i][j] += (b[i+2][j+2]);
      }}
 multiple2x2(temp,temp2,t1);
 printf("T Matrix\n");
 display(t1);
 //U
 for ( i = 0; i < 2; ++i) {
   for ( j = 0; j < 2; ++j) {
         temp[i][j] = 0;
         temp2[i][j] = 0;
         temp[i][j] += ((a[i+2][j]) - (a[i][j]));
         temp2[i][j] += (b[i][j] + b[i][j+2]);
      }}
 multiple2x2(temp,temp2,u1);
 printf("U Matrix\n");
 display(u1);
 //V
 for ( i = 0; i < 2; ++i) {
   for ( j = 0; j < 2; ++j) {
         temp[i][j] = 0;
         temp2[i][j] = 0;
         temp[i][j] += ((a[i][j+2]) - (a[i+2][j+2]));
         temp2[i][j] += (b[i+2][j] + b[i+2][j+2]);
      }}
 multiple2x2(temp,temp2,v1);
 printf("V Matrix\n");
 display(v1);
//Final Matrix Calculation
for ( i = 0; i < 2; ++i) {
   for ( j = 0; j < 2; ++j) {
      //c11
      c[i][j] = ((p1[i][j] + s1[i][j]) - t1[i][j]) + v1[i][j];
      //c12
      c[i][j+2] = r1[i][j] + t1[i][j];
      //c21
```

```
          c[i+2][j] = q1[i][j] + s1[i][j];
          //c22
          c[i+2][j+2] = ((p1[i][j] + r1[i][j]) - q1[i][j]) + u1[i][j];
          }}
   printf("\nC Matrix\n");
   for ( i = 0; i < 4; ++i) {
      printf("|");
      for ( j = 0; j < 4; ++j) {
            printf("%d ",c[i][j]);
         }
      printf("|");
      printf("\n");
      }
    return 0;
}
void display(int z[2][2]){
    int i, j;
    for(i = 0; i < 2; i++){
        printf("|");
        for(j = 0; j < 2; j++){
             printf("%d ",z[i][j]);
         }
        printf("|");
        printf("\n");
    }}
void multiple2x2(int temp[2][2],int temp2[2][2], int z[2][2]){
    int p,q,r,s,t,u,v;
    p = (temp[0][0] + temp[1][1]) * (temp2[0][0] + temp2[1][1]);
    q = (temp[1][0] + temp[1][1]) * (temp2[0][0]);
    r = temp[0][0] * (temp2[0][1] - temp2[1][1]);
    s = temp[1][1] * (temp2[1][0] - temp2[0][0]);
    t = (temp[0][0] + temp[0][1]) * (temp2[1][1]);
    u = (temp[1][0] - temp[0][0]) * (temp2[0][0] + temp2[0][1]);
    v = (temp[0][1] - temp[1][1]) * (temp2[1][0] + temp2[1][1]);
```

```
    z[0][0] = p + s - t + v;
    z[0][1] = r + t;
    z[1][0] = q + s;
    z[1][1] = p + r - q + u;
}
```

Output :

```
A Matrix
-2 5 -8 9
6 -7 4 3
1 -8 6 7
-9 5 3 4
B Matrix
5 -3 8 6
-4 2 9 1
-9 3 -4 7
8 -5 2 6
P Matrix
|-20 112 |
|15 12 |
Q Matrix
|39 -23 |
|-66 36 |
R Matrix
|11 -23 |
|23 29 |
S Matrix
|0 -13 |
|6 -10 |
T Matrix
|68 14 |
|-48 46 |
```

```
U Matrix
|-26 -30 |
|-135 -9 |
V Matrix
|202 -138 |
|-23 9 |

C Matrix
|114 -53 79 -9 |
|46 -35 -25 75 |
|39 -36 -74 82 |
|-60 26 -31 -4 |
```

D2) Strassen’s Matrix Multiplication (2x2)

```
#include<stdio.h>
#include<stdlib.h>
int a[2][2], b[2][2];
int c[2][2];
int main()
{
    int i,j,p,q,r,s,t,u,v;
    printf("A matrix\n");
    for (i = 0; i < 2; i++){
        for (j = 0; j < 2; j++){
            scanf("%d",&a[i][j]);
        } }
    printf("B matrix\n");
    for (i = 0; i < 2; i++){
        for (j = 0; j < 2; j++){
            scanf("%d",&b[i][j]);
        }}
    p = (a[0][0] + a[1][1]) * (b[0][0] + b[1][1]);
    q = (a[1][0] + a[1][1]) * (b[0][0]);
    r = a[0][0] * (b[0][1] - b[1][1]);
    s = a[1][1] * (b[1][0] - b[0][0]);
    t = (a[0][0] + a[0][1]) * (b[1][1]);
    u = (a[1][0] - a[0][0]) * (b[0][0] + b[0][1]);
    v = (a[0][1] - a[1][1]) * (b[1][0] + b[1][1]);
    printf("\np = %d",p);
    printf("\nq = %d",q);
    printf("\nr = %d",r);
    printf("\ns = %d",s);
    printf("\nt = %d",t);
    printf("\nu = %d",u);
    printf("\nv = %d\n\n",v);
    c[0][0] = p + s - t + v;
    c[0][1] = r + t;
```

```
    c[1][0] = q + s;
    c[1][1] = p + r - q + u;
    printf("C Matrix\n");
    for (i = 0; i < 2; i++){
        printf("|");
        for (j = 0; j < 2; j++){
            printf("%d ",c[i][j]);
        }
        printf("|");
        printf("\n");
    }
    return 0;
}
```

Output :

```
A matrix
14 -13
-9 8
B matrix
-17 11
-15 -20
p = -814
q = 17
r = 434
s = 16
t = -20
u = 138
v = 735

C Matrix
|-43 414 |
|33 -259 |
```