1.gcd of 3 numbers

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
int gcdEuclid(int m, int n)
{
   int r;
   while(n!=0)
   {
      r=m%n;
      m=n;
      n=r;
   }
   return m;
}
int gcdConsec(int m, int n)
{
   int t;
   if(m<n)
      t=m;
   else
      t=n;
   while(t>0)
   {
      if(m%t==0)
      {
         if(n%t==0)
            return t;
         else
            t--;
```

```
        }
        else
            t--;
    }
}
int gcdMidSch(int m, int n)
{
    int gcd=1,a=2;
    while(a<=m && a<=n && m!=0 && n!=0)
    {
        if(m%a==0)
        {
            if(n%a==0)
            {
                gcd=gcd*a;
                n=n/a;
            }
            m=m/a;
        }
        else
            a++;
    }
    return gcd;
}
int main()
{
    int m,n,r,t,ch,g,gcd;
    clock_t start,end;
    double time;
    printf("\nenter the value of m\n");
    scanf("%d",&m);
```

```
printf("\nenter the value of n\n");
scanf("%d",&n);
if(m==0 || n==0)
{
   printf("enter a value grater than 0");
   exit(0);
}
while(1)
{
   printf("enter your choice\n");
   printf("\n1.euclid\n2.consecutive\n3.middle school\npress any key to exit");
   scanf("%d",&ch);
   switch(ch)
   {
      case 1:start=clock();
            g=gcdEuclid(m,n);
            end=clock();
            time=(double)(end-start)/CLOCKS_PER_SEC;
            printf("\neuclid's algorithum:gcd(%d,%d)=%d",m,n,g);
            printf("\ntime taken=%f seconds",time);
            break;
       case 2:start=clock();
            g=gcdConsec(m,n);
            end=clock();
            time=(double)(end-start)/CLOCKS_PER_SEC;
            printf("\nconsecutive integer:gcd(%d,%d)=%d",m,n,g);
            printf("\ntime taken=%f seconds",time);
            break;
       case 3:start=clock();
            g=gcdMidSch(m,n);
            end=clock();
```

```
            time=(double)(end-start)/CLOCKS_PER_SEC;
            printf("\nmiddle school:gcd(%d,%d)=%d",m,n,g);
            printf("\ntime taken=%f seconds",time);
            break;
        default:exit(0);
      }
   }
   return 0;
}
```

2.sieve of erasthones

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void sieve(int n)
{

   int p,a[n+1],j;
   for(p=2;p<=n;p++)
      a[p]=p;
   for(p=2;p<=floor(sqrt(n));p++)
   {
      if(a[p]!=0)
      {
         j=p*p;
         while(j<=n)
         {
            a[j]=0;
            j=j+p;
         }
```

```
        }
    }
    int i=0,L[n+1];
    for(p=2;p<=n;p++)
    {
        if(a[p]!=0)
        {
            L[i]=a[p];
            i++;
        }
    }
    printf("\nthe prime numbers from 2 to %d are\n",n);
    for(j=0;j<i;j++)
    printf("%d\t",L[j]);
}
int main()
{
    int n;
    clock_t start,end;
    double time;
    printf("enter the value of n\n");
    scanf("%d",&n);
    if(n<2)
    {
        printf("\nvalue of n must be greater than 2");
        exit(0);
    }
    start=clock();
    sieve(n);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
```

```
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```

3.sequential search

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int sequentialSearch(int a[],int k,int n)
{
    int i;
    i=0;
    while(i<n && a[i]!=k)
        i++;
    if(i<n)
        return i;
    else
        return -1;
}
int main()
{
    int a[100],k,n,i;
    clock_t start,end;
    double time;
    printf("\nenter the size of the array\n");
    scanf("%d",&n);
    printf("\nenter the array elements\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\nenter the key position\n");
```

```
    scanf("%d",&k);
    start=clock();
    int pos=sequentialSearch(a,k,n);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    if(pos==-1)
        printf("\nkey %d is not found in the array",k);
    else
        printf("\nkey %d is found in the position %d in the array",k,pos);
    printf("\ntime taken=%f seconds",time);
}
```

4.minimum and maximum element

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int maxelement(int a[],int n)
{
    int maxval;
    int i=0;
    maxval=a[0];
    for(i=1;i<n;i++)
    {
        if(a[i]>maxval)
            maxval=a[i];
    }
    return maxval;
}
int minelement(int a[],int n)
{
```

```
    int minval;
    int i=0;
    minval=a[0];
    for(i=1;i<n;i++)
    {
        if(a[i]<minval)
            minval=a[i];
    }
    return minval;
}
int main()
{
    int a[100],n,i;
    clock_t start,end;
    double time;
    printf("enter the size of the array\n");
    scanf("%d",&n);
    printf("\nenter the array elements\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    start=clock();
    int minv=minelement(a,n);
    int maxv=maxelement(a,n);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    printf("\nminimum element is=%d\n",minv);
    printf("\nmaximum element is=%d\n",maxv);
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```

5.array elements are unique or not

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
#include<stdbool.h>
bool elementUniqueness(int a[],int n)
{
   int i=0,j=0;
   for(i=0;i<n-1;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(a[i]==a[j])
            return false;
      }
   }
   return true;
}
int main()
{
   int a[100],n,i;
   clock_t start,end;
   double time;
   printf("enter the size of the array\n");
   scanf("%d",&n);
   printf("\nenter the array elements\n");
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   start=clock();
   bool res=elementUniqueness(a,n);
```

```
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    if(res==false)
        printf("\narray elements are not unique");
    else
        printf("\narray elements are unique");
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```

6.bubble sort

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void bubbleSort(int a[],int n)
{
    int temp,i,j;
    for(i=0;i<=n-2;i++)
    {
        for(j=0;j<=n-2;j++)
        {
            if(a[j+1]<a[j])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
}
```

```
int main()
{
   int a[100],n,i;
   clock_t start,end;
   double time;
   printf("enter the size of the array\n");
   scanf("%d",&n);
   printf("\nenter the array elements\n");
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   start=clock();
   bubbleSort(a,n);
   end=clock();
   time=(double)(end-start)/CLOCKS_PER_SEC;
   printf("\nsorted array are\n");
   for(i=0;i<n;i++)
      printf("%d\t",a[i]);
   printf("\ntime taken = %f seconds",time);
   return 0;
}
```

7.selection sort

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void selectionSort(int a[],int n)
{
   int i,j,temp,min;
   for(i=0;i<=n-2;i++)
   {
```

```
        min=i;
        for(j=i+1;j<=n;j++)
        {
           if(a[j]<a[min])
              min=j;
        }
        temp=a[i];
        a[i]=a[min];
        a[min]=temp;
    }
}
int main()
{
    int a[100],n,i;
    clock_t start,end;
    double time;
    printf("\nenter the size of the array\n");
    scanf("%d",&n);
    printf("\nenter the array elements\n");
    for(i=0;i<n;i++)
       scanf("%d",&a[i]);
    start=clock();
    selectionSort(a,n);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    printf("\nsorted arrays are\n");
    for(i=0;i<n;i++)
       printf("%d\t",a[i]);
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```

8.BFStringMatching

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int BFStringMatching(char t[],char p[],int m,int n)
{
   int i,j;
   for(i=0;i<n-m;i++)
   {
      while(j<m && t[i+j]==p[j])
         j++;
      if(j==m)
         return i;
   }
   return -1;
}
int main()
{
   char t[100],p[100];
   int m,n;
   clock_t start,end;
   double time;
   printf("\nenter the text string\n");
   gets(t);
   printf("\nenter the pattern string\n");
   gets(p);
   n=strlen(t);
   m=strlen(p);
   if(n<m)
   {
```

```
        printf("\ntext string must be larger than pattern string");
        exit(0);
    }
    start=clock();
    int pos=BFStringMatching(t,p,m,n);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    if(pos==-1)
        printf("\npattern string is not found in the text siring");
    else
        printf("\npattern string is found in the text string at position %d",pos);
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```

9.Binary Search

```
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<time.h>
int binarySearch(int a[],int n,int k)
{
    int l,r,m;
    l=0;
    r=n-1;
    while(l<=r)
    {
        m=floor((l+r)/2);
        if(a[m]==k)
            return m;
        else if(a[m]>k)
```

```
            r=m-1;
        else
            l=m+1;
    }
    return -1;
}
int main()
{
    int a[100],k,n,i;
    clock_t start,end;
    double time;
    printf("\nenter the size of the array\n");;
    scanf("%d",&n);
    printf("\nenter the array elements\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
    printf("\nenter the key\n");
    scanf("%d",&k);
    start=clock();
    int pos=binarySearch(a,n,k);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    if(pos==-1)
        printf("\nkey %d is not found in the array",k);
    else
        printf("\nkey %d is found in the position %d in the array",k,pos);
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```

10.Insertion Sort

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>
void insertionSort(int a[],int n)
{
   for(int i=1;i<n;i++)
   {
      int v=a[i];
      int j=i-1;
      while(j>=0 && a[j]>v)
      {
         a[j+1]=a[j];
         j--;
      }
      a[j+1]=v;
   }
}
int main()
{
   int a[100],n,i;
   clock_t start,end;
   double time;
   printf("enter the size of the array\n");
   scanf("%d",&n);
   printf("\nenter the array elements\n");
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   start=clock();
   insertionSort(a,n);
```

```
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    printf("\nsorted array is\n");
    for(i=0;i<n;i++)
        printf("%d\t",a[i]);
    printf("time taken=%f seconds",time);
    return 0;
}
```

11.DFS Graph

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int count=0,visited[10]={0},n,a[10][10];
void dfs(int v)
{
    int w;
    count++;
    visited[v]=count;
    for(w=1;w<=n;w++)
        if(a[v][w]==1 && visited[w]==0)
        dfs(w);
}
int main()
{
    int i,j;
    printf("enter the number of vertices");
    scanf("%d",&n);
    printf("enter the adjacency matrix");
    for(i=1;i<n+1;i++)
```

```
    for(j=1;j<n+1;j++)
    scanf("%d",&a[i][j]);
  clock_t start,end;
  double time;
  start=clock();
  dfs(1);
  end=clock();
  time=(double)(end-start)/CLOCKS_PER_SEC;
  if(count==n)
    printf("the graph is connected");
  else
    printf("the graph is not connected");
  printf("time taken=%f seconds",time);
  return(0);
}
```

12.topological sort

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int count=0,visited[10]={0},n,a[10][10],k=0,pop[10];
void dfs(int v)
{
  int w;
  count++;
  visited[v]=count;
  for(w=1;w<=n;w++)
    if(a[v][w]==1 && visited[w]==0)
    dfs(w);
  pop[k++]=v;
```

```
}
int main()
{
   int i,j,v;
   printf("\nenter the number of vertices\n");
   scanf("%d",&n);
   printf("\nenter the adjacency matrix\n");
   for(i=1;i<n+1;i++)
      for(j=1;j<n+1;j++)
      scanf("%d",&a[i][j]);
   clock_t start,end;
   double time;
   start=clock();
   for(v=1;v<=n;v++)
   {
      if(visited[v]==0)
         dfs(v);
   }
   end=clock();
   time=(double)(end-start)/CLOCKS_PER_SEC;
   printf("\nthe topological sorting is\n");
   for(i=k-1;i>=0;i--)
      printf("%d\t",pop[i]);
   printf("\ntime taken=%f seconds",time);
   return(0);
}
```

13.BFS Traversal

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

```
#include<time.h>
int count=0,visited[10]={0},n,a[10][10];
void bfs(int v)
{
   int w;
   count++;
   visited[v]=count;
   int q[10],front;
   int f=0,r=-1;
   q[++r]=v;
   while(f<=r)
   {
      front=q[f];
      for(w=1;w<=n;w++)
      {
         if(a[front][w]==1 && visited[w]==0)
         {
            count++;
            visited[w]=count;
            printf("%d\t",w);
            q[++r]=w;
         }
      }
      f++;

   }
}

int main()
{
   int i,j,s;
```

```
    printf("\nenter the number of vertices\n");
    scanf("%d",&n);
    printf("\nenter the adjacency matrix\n");
    for(i=1;i<n+1;i++)
       for(j=1;j<n+1;j++)
       scanf("%d",&a[i][j]);
    printf("\nenter the starting node\n");
    scanf("%d",&s);
    clock_t start,end;
    double time;
    start=clock();
    printf("\nthe nodes reachable are\n");
    bfs(s);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    printf("\ntime taken=%f seconds",time);
    return(0);
}
```

14.Merge Sort

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
void merge(int b[],int c[],int a[],int p, int q)
{
    int i=0,j=0,k=0,m=0;
    while(i<p && j<q)
    {
       if(b[i]<=c[j])
       {
```

```
            a[k]=b[i];
            i++;
        }
        else
        {
            a[k]=c[j];
            j++;
        }
        k++;
    }
    if(i==p)
    {
        for(m=j;m<=q-1;m++)
        {
            a[k]=c[m];
            k++;
        }
    }
    else
    {
        for(m=i;m<=p-1;m++)
        {
            a[k]=b[m];
            k++;
        }
    }
}
void mergeSort(int a[],int n)
{
    int b[100],c[100],i,p=0,q=0;
    if(n>1)
```

```
   {
      for(i=0;i<=floor(n/2)-1;i++)
      {
         b[p]=a[i];
         p++;
      }
      for(i=floor(n/2);i<=n-1;i++)
      {
         c[q]=a[i];
         q++;
      }
      mergeSort(b,p);
      mergeSort(c,q);
      merge(b,c,a,p,q);
   }
}
int main()
{
int i,n,a[100];
clock_t start,end;
double time;
printf("\nenter the size of the array\n");
scanf("%d",&n);
printf("\nenter the array elements\n");
for(i=0;i<n;i++)
   scanf("%d",&a[i]);
start=clock();
mergeSort(a,n);
end=clock();
time=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nsorted array is\n");
```

```
for(i=0;i<n;i++)
   printf("%d\t",a[i]);
printf("time taken=%f seconds",time);
return 0;
}
```

15.Quick Sort

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>
int partition(int a[], int l ,int r)
{
    int p,i,j,temp;
    p=a[l];
    i=l;
    j=r+1;
    do
    {
       do
       {
          i=i+1;
       }
       while(a[i]<p && p<=r);
       do
       {
          j=j-1;
       }
       while(a[j]>p);
       temp=a[i];
       a[i]=a[j];
```

```
        a[j]=temp;
    }
    while(i<j);
      temp=a[i];
      a[i]=a[j];
      a[j]=temp;
      temp=a[l];
      a[l]=a[j];
      a[j]=temp;
      return j;
}
void quickSort(int a[],int l,int r)
{
   int s;
   if(l<r)
   {
      s=partition(a,l,r);
      quickSort(a,l,s-1);
      quickSort(a,s+1,r);
   }
}
int main()
{
int i,n,a[100];
clock_t start,end;
double time;
printf("\nenter the size of the array\n");
scanf("%d",&n);
printf("\nenter the array elements\n");
for(i=0;i<n;i++)
   scanf("%d",&a[i]);
```

```
start=clock();
quickSort(a,0,n-1);
end=clock();
time=(double)(end-start)/CLOCKS_PER_SEC;
printf("\nsorted array is::\n");
for(i=0;i<n;i++)
   printf("%d\t",a[i]);
printf("time taken=%f seconds",time);
return 0;
}
```

16.Strassens multiplication

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include<time.h>
void strassens(int a[][2],int b[][2],int c[][2])
{
   int m1,m2,m3,m4,m5,m6,m7;
   m1=(a[0][0]+a[1][1])*(b[0][0]+b[1][1]);
   m2=(a[1][0]+a[1][1])*b[0][0];
   m3=a[0][0]*(b[0][1]-b[1][1]);
   m4=a[1][1]*(b[1][0]-b[0][0]);
   m5=(a[0][0]+a[0][1])*b[1][1];
   m6=(a[1][0]-a[0][0])*(b[0][0]+b[0][1]);
   m7=(a[0][1]-a[1][1])*(b[1][0]+b[1][1]);
   c[0][0]=m1+m4-m5+m7;
   c[0][1]=m3+m5;
   c[1][0]=m2+m4;
   c[1][1]=m1+m3-m2+m6;
```

```
}
int main()
{
   int a[2][2],b[2][2],c[2][2],i,j;
   clock_t start,end;
   double time;
   printf("Enter the matrix A\n");
   for(i=0;i<2;i++)
   {
      for(j=0;j<2;j++)
      {
         scanf("%d",&a[i][j]);
      }
   }
   printf("\nEnter the matrix B\n");
   for(i=0;i<2;i++)
   {
      for(j=0;j<2;j++)
      {
         scanf("%d",&b[i][j]);
      }
   }
   start=clock();
   strassens(a,b,c);
   end=clock();
   time=(double)(end-start)/CLOCKS_PER_SEC;
   printf("\nResultant matrix C\n");
   for(i=0;i<2;i++)
   {
      for(j=0;j<2;j++)
      {
```

```
            printf("%d\t",c[i][j]);
        }
        printf("\n");
    }
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```

17.Horsepool matching

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>
#include<time.h>
int table[256];
void shiftable(char p[],int m)
{
    int i,j;
    for(i=0;i<255;i++)
        table[i]=m;
    for(j=0;j<=m-2;j++)
        table[p[j]]=m-1-j;
}
int horsepoolmethod(char t[],char p[],int m,int n)
{
    int i,j;
    shiftable(p,m);
    i=m-1;
    while(i<=n-1)
    {
        j=0;
```

```
        while(j<m && t[i-j]==p[m-1-j])
        {
           j=j+1;
        }
        if(j==m)
           return i-m+1;
        else
           i=i+table[t[i]];
    }
    return -1;
}
int main()
{
char t[100],p[100];
int m,n;
clock_t start,end;
double time;
printf("\nenter the text string\n");
gets(t);
printf("\nenter the pattern string\n");
gets(p);
n=strlen(t);
m=strlen(p);
if(n<m)
{
    printf("\ntext string should be larger then pattern string");
    exit(0);
}
start=clock();
int pos=horsepoolmethod(t,p,m,n);
end=clock();
```

```
time=(double)(end-start)/CLOCKS_PER_SEC;
if(pos==-1)
   printf("\npattern is not found\n");
else
   printf("\npattern is found in the position %d",pos);
printf("\ntime taken=%f seconds",time);
return 0;
}
```

17. Heap Sort

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
#include<math.h>
#include<time.h>
void bottomupheap(int a[],int n)
{
   int v,i,j,k;
   bool heap;
   for(i=floor(n/2);i>=1;i--)
   {
      i=k;
      v=a[k];
      heap=false;
      while(!heap && 2*k<=n)
      {
         j=2*k;
         if(j<n)
         {
            if((a[j]<a[j+1]))
               j=j+1;
```

```
            }
            if(v>=a[j])
                heap=true;
            else
            {
                a[k]=a[j];
                k=j;
            }
        }
        a[k]=v;
    }
}
void heapsort(int a[],int n)
{
    int temp;
    bottomupheap(a,n);
    while(n>1)
    {
        temp=a[1];
        a[1]=a[n];
        a[n]=temp;
        n=n-1;
        bottomupheap(a,n);
    }
}
int main()
{
    int a[100],n,i;
    clock_t start,end;
    double time;
    printf("\nenter the size of the array\n");
```

```
    scanf("%d",&n);
    printf("\nenter the array elements\n");
    for(i=1;i<=n;i++)
       scanf("%d\n",&a[i]);
    start=clock();
    heapsort(a,n);
    end=clock();
    time=(double)(end-start)/CLOCKS_PER_SEC;
    printf("\n sorted array\n");
    for(i=1;i<=n;i++)
       printf("%d\t",a[i]);
    printf("\ntime taken=%f seconds",time);
    return 0;
}
```