Here are each program again, now with a sample run (user input → program output) so you can see how it works.

---

### 1. GCD of two numbers

#### a) Euclid’s Method

```c

#include <stdio.h>

int gcd(int a, int b){ return b==0 ? a : gcd(b, a%b); }

int main(){

int m, n;

printf("Enter two numbers: ");

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

printf("GCD (Euclid) = %d\n", gcd(m,n));

return 0;

}

```

\*\*Example:\*\*

```

Enter two numbers: 18 30

GCD (Euclid) = 6

```

#### b) Consecutive Integer Checking

```c

#include <stdio.h>

int gcd\_ci(int a, int b){

int g = (a<b?a:b);

for(; g>0; g--) if(a%g==0 && b%g==0) break;

return g;

}

int main(){

int m, n;

printf("Enter two numbers: ");

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

printf("GCD (Consecutive) = %d\n", gcd\_ci(m,n));

return 0;

}

```

\*\*Example:\*\*

```

Enter two numbers: 18 30

GCD (Consecutive) = 6

```

#### c) Middle School Method

```c

#include <stdio.h>

int gcd\_ms(int a, int b){

int g=1, i=2;

while(i<=a && i<=b){

if(a%i==0 && b%i==0){ g\*=i; a/=i; b/=i; }

else i++;

}

return g;

}

int main(){

int m, n;

printf("Enter two numbers: ");

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

printf("GCD (Middle School) = %d\n", gcd\_ms(m,n));

return 0;

}

```

\*\*Example:\*\*

```

Enter two numbers: 18 30

GCD (Middle School) = 6

```

---

### 2. Linear Search (with timing)

```c

#include <stdio.h>

#include <time.h>

int main(){

int n, key, pos=-1;

printf("Enter list size n: ");

scanf("%d", &n);

int a[n];

for(int i=0;i<n;i++){

printf("a[%d]= ", i);

scanf("%d", &a[i]);

}

printf("Enter key to search: ");

scanf("%d", &key);

clock\_t t0 = clock();

for(int i=0;i<n;i++)

if(a[i]==key){ pos=i; break; }

double dt = (double)(clock()-t0)/CLOCKS\_PER\_SEC;

if(pos>=0) printf("Found at index %d\n", pos);

else printf("Not found\n");

printf("Time: %f sec\n", dt);

return 0;

}

```

\*\*Example:\*\*

```

Enter list size n: 5

a[0]= 4

a[1]= 7

a[2]= 1

a[3]= 9

a[4]= 3

Enter key to search: 9

Found at index 3

Time: 0.000002 sec

```

---

### 3. Largest Element (with timing)

```c

#include <stdio.h>

#include <time.h>

int main(){

int n;

printf("Enter list size n: ");

scanf("%d", &n);

int a[n];

for(int i=0;i<n;i++){

printf("a[%d]= ", i);

scanf("%d", &a[i]);

}

clock\_t t0 = clock();

int max = a[0];

for(int i=1;i<n;i++)

if(a[i]>max) max = a[i];

double dt = (double)(clock()-t0)/CLOCKS\_PER\_SEC;

printf("Largest element = %d\nTime: %f sec\n", max, dt);

return 0;

}

```

\*\*Example:\*\*

```

Enter list size n: 4

a[0]= 12

a[1]= 5

a[2]= 19

a[3]= 7

Largest element = 19

Time: 0.000001 sec

```

---

### 4. String Matching (Brute Force)

```c

#include <stdio.h>

#include <string.h>

int main(){

char txt[500], pat[100];

printf("Enter text: ");

fgets(txt,500,stdin);

printf("Enter pattern: ");

fgets(pat,100,stdin);

int n=strlen(txt)-1, m=strlen(pat)-1, found=0;

for(int i=0;i<=n-m;i++){

int j=0;

while(j<m && txt[i+j]==pat[j]) j++;

if(j==m){ printf("Pattern at index %d\n", i); found++; }

}

if(!found) printf("No match\n");

return 0;

}

```

\*\*Example:\*\*

```

Enter text: hello world

Enter pattern: or

Pattern at index 7

```

---

### 5. Selection Sort (with timing)

```c

#include <stdio.h>

#include <time.h>

void selsort(int a[], int n){

for(int i=0;i<n-1;i++){

int min=i;

for(int j=i+1;j<n;j++)

if(a[j]<a[min]) min=j;

int t=a[i]; a[i]=a[min]; a[min]=t;

}

}

int main(){

int n;

printf("Enter n: ");

scanf("%d",&n);

int a[n];

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

clock\_t t0=clock();

selsort(a,n);

double dt=(double)(clock()-t0)/CLOCKS\_PER\_SEC;

printf("Sorted: ");

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

printf("\nTime: %f sec\n", dt);

return 0;

}

```

\*\*Example:\*\*

```

Enter n: 5

5 2 9 1 6

Sorted: 1 2 5 6 9

Time: 0.000003 sec

```

---

### 6. Insertion Sort (with timing)

```c

#include <stdio.h>

#include <time.h>

void insort(int a[], int n){

for(int i=1;i<n;i++){

int key=a[i], j=i-1;

while(j>=0 && a[j]>key){

a[j+1]=a[j]; j--;

}

a[j+1]=key;

}

}

int main(){

int n; printf("Enter n: "); scanf("%d",&n);

int a[n];

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

clock\_t t0=clock();

insort(a,n);

double dt=(double)(clock()-t0)/CLOCKS\_PER\_SEC;

printf("Sorted: ");

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

printf("\nTime: %f sec\n", dt);

return 0;

}

```

\*\*Example:\*\*

```

Enter n: 4

8 3 5 1

Sorted: 1 3 5 8

Time: 0.000002 sec

```

---

### 7. Merge Sort (with timing)

```c

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void merge(int a[],int l,int m,int r){

int n1=m-l+1, n2=r-m, L[n1], R[n2];

for(int i=0;i<n1;i++) L[i]=a[l+i];

for(int j=0;j<n2;j++) R[j]=a[m+1+j];

int i=0,j=0,k=l;

while(i<n1 && j<n2)

a[k++] = (L[i]<=R[j]) ? L[i++] : R[j++];

while(i<n1) a[k++]=L[i++];

while(j<n2) a[k++]=R[j++];

}

void msort(int a[],int l,int r){

if(l<r){

int m=(l+r)/2;

msort(a,l,m);

msort(a,m+1,r);

merge(a,l,m,r);

}

}

int main(){

int n; printf("Enter n: "); scanf("%d",&n);

int \*a=malloc(n\*sizeof(int));

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

clock\_t t0=clock();

msort(a,0,n-1);

double dt=(double)(clock()-t0)/CLOCKS\_PER\_SEC;

printf("Sorted: ");

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

printf("\nTime: %f sec\n", dt);

free(a);

return 0;

}

```

\*\*Example:\*\*

```

Enter n: 6

7 2 9 4 1 5

Sorted: 1 2 4 5 7 9

Time: 0.000005 sec

```

---

### 8. Quick Sort (with timing)

```c

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int part(int a[], int l, int h){

int p = a[h], i = l-1;

for(int j=l;j<h;j++){

if(a[j]<p){ i++; int t=a[i]; a[i]=a[j]; a[j]=t; }

}

int t=a[i+1]; a[i+1]=a[h]; a[h]=t;

return i+1;

}

void qsort\_rec(int a[], int l, int h){

if(l<h){

int pi=part(a,l,h);

qsort\_rec(a,l,pi-1);

qsort\_rec(a,pi+1,h);

}

}

int main(){

int n; printf("Enter n: "); scanf("%d",&n);

int \*a=malloc(n\*sizeof(int));

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

clock\_t t0=clock();

qsort\_rec(a,0,n-1);

double dt=(double)(clock()-t0)/CLOCKS\_PER\_SEC;

printf("Sorted: ");

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

printf("\nTime: %f sec\n", dt);

free(a);

return 0;

}

```

\*\*Example:\*\*

```

Enter n: 5

3 8 1 6 4

Sorted: 1 3 4 6 8

Time: 0.000003 sec

```

---

Compile (`gcc prog.c -o prog`) and run (`./prog`), then enter the sample inputs to see the matching outputs.