1. Program to implement the Bubble sort .

```
#include<stdio.h>
#include<time.h>
void bubble_sort(int [],int);
void main()
{
 int i,n,a[10];
 printf("Enter size:\n");
 scanf("%d",&n);
 printf("Enter array elements:\n");
 for(i=0;i<n;i++)
   scanf("%d",&a[i]);
 bubble_sort(a,n);
 }
 void bubble_sort(int a[],int n)
 {
    clock_t start_t, end_t, total_t;
   start_t = clock();
   printf("Starting of the function, start_t = %ld\n", start_t);
 int t,i,j;
 for(i=1;i<n;i++)
 {
  for(j=0;j<n-i;j++)
  {
   if(a[j]>a[j+1])
   {
    t=a[j];
    a[j]=a[j+1];
    a[j+1]=t;
    }
    }
    }
    printf("\nIn Ascending order:");
    for(i=0;i<n;i++)
       printf("%d\t",a[i]);
       end_t = clock();
  printf("\nEnd of the function, end_t = %ld\n", end_t);
  total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
  printf("Total time taken by CPU: %d\n", total_t  );
   printf("Exiting of the program...\n");
    }
```

2. Program to implement selection sort

```
#include<stdio.h>
#include <time.h>
void selection_sort(int, int[]);
void main()
{
 int i,n,a[10];
 printf("Enter size:\n");
 scanf("%d",&n);
 printf("Enter array elements:\n");
 for(i=0;i<n;i++)
   scanf("%d",&a[i]);

 selection_sort(n,a);
 }


void selection_sort(int n, int a[])
{
   clock_t start_t, end_t, total_t;
   start_t = clock();
   printf("Starting of the function, start_t = %ld\n", start_t);
   int i, j, min;

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

      // Swap the found minimum element with the first element
        if(min != i)

         {
         int t=a[min];
         a[min]=a[i];
         a[i]=t;
   }
   }
   printf("\nIn ascending order:");
   for(i=0;i<n;i++)
      printf("%d\t",a[i]);
              end_t = clock();
  printf("\nEnd of the  function, end_t = %ld\n", end_t);
  total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
  printf("Total time taken by CPU: %d\n", total_t  );
   printf("Exiting of the program...\n");
}
```

3. Program to implement the linear search

```
#include<stdio.h>

#include<time.h>
void linear_search(int,int,int[]);
void main()
{
   int i,n,a[10],key;
   printf("Enter size:\n");
   scanf("%d",&n);
   printf("Enter array elements:\n");
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);
   printf("\nEnter key:\n");
   scanf("%d",&key);
   linear_search(n,key,a);
}
void linear_search(int n,int key,int a[])
{
   int i,f=0;
   clock_t start_t, end_t, total_t;
   start_t = clock();
   printf("Starting of the linear search function, start_t = %ld\n", start_t);
   for( i=0;i<n;i++)
   {


      if(key==a[i])
      {
         printf("Successful search\n");
         f=1;
         break;
      }
 }
 if(f==0)
 printf("Unsuccessful search");
 end_t = clock();
 printf("End of the linear search function, end_t = %ld\n", end_t);
 total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
 printf("Total time taken by CPU: %d\n", total_t  );
  printf("Exiting of the program...\n");
}
```

4. Program to implement the binary search

```
#include<stdio.h>
#include <time.h>
void binary_search(int,int,int[]);
void main()
{
 int i,n,key,a[10];
 printf("Enter size:\n");
 scanf("%d",&n);
 printf("Enter array elements:");
 for(i=0;i<n;i++)
   scanf("%d",&a[i]);
 printf("\nEnter key:");
 scanf("%d",&key);
 binary_search(n,key,a);
 }
 void binary_search(int n,int key, int a[])
 {
     clock_t start_t, end_t, total_t;
   start_t = clock();
   printf("Starting of the function, start_t = %ld\n", start_t);
 int t,i,j;
 for(i=1;i<n;i++)
 {
  for(j=0;j<n-i;j++)
  {
   if(a[j]>a[j+1])
   {
    t=a[j];
    a[j]=a[j+1];
    a[j+1]=t;
    }
    }
    }
 int h=n-1,l=0,mid,flag=0;
 while(h>=l)
 {
  mid=(h+l)/2;
  if(a[mid]==key)
  {
   flag=1;
   printf("Successful search");
   break;
   }
   if(a[mid]<key)
   {
    l=mid+1;
    }
    if(a[mid]>key)
    {
    h=mid-1;
    }
    }
    if(flag==0)
       printf("Unsuccessful search");
           end_t = clock();
 printf("\nEnd of the function, end_t = %ld\n", end_t);
 total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
 printf("Total time taken by CPU: %d\n", total_t  );
  printf("Exiting of the program...\n");
     }
```