**Assignment – 2**

**Q1. Implementation of Bubble Sort, Insertion Sort and Selection Sort with random values and calculate the time taken.**

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
#define n 10000  
  
int\* create\_array(int, int);  
int\* copy(int\*);  
void display\_array(int\*);  
void bubble\_sort(int\*);  
void insertion\_sort(int\*);  
void selection\_sort(int\*);  
void display\_time(char[], clock\_t, clock\_t);  
  
int main() {  
 srand(time(NULL));  
 clock\_t start, end;  
 int low, high;  
 printf("Enter the range of random numbers to generate: ");  
 scanf("%d%d", &low, &high);  
 int \*array = create\_array(low, high), \*copy\_array;  
 printf("Array before sorting:\n");  
 display\_array(array);  
  
 copy\_array = copy(array);  
 start = clock();  
 bubble\_sort(copy\_array);  
 end = clock();  
 printf("Array after sorting using bubble sort:\n");  
 display\_array(copy\_array);  
 free(copy\_array);  
 display\_time("bubble sort", start, end);  
  
 copy\_array = copy(array);  
 start = clock();  
 insertion\_sort(copy\_array);  
 end = clock();  
 printf("Array after sorting using insertion sort:\n");  
 display\_array(copy\_array);  
 free(copy\_array);  
 display\_time("insertion sort", start, end);  
  
 copy\_array = copy(array);  
 start = clock();  
 selection\_sort(copy\_array);  
 end = clock();  
 printf("Array after sorting using selection sort:\n");  
 display\_array(copy\_array);  
 free(copy\_array);  
 display\_time("selection\_sort", start, end);  
  
 free(array);  
 return 0;  
}  
  
int\* create\_array(int low, int high) {  
 int \*array = (int\*)malloc(sizeof(int) \* n), i;  
  
 for (i = 0; i < n; i++) {  
 array[i] = low + rand() % (high - low + 1);  
 }  
 return array;  
}  
  
int\* copy(int\* array) {  
 int \*res = (int\*)malloc(sizeof(int) \* n), i;  
  
 for (i = 0; i < n; i++) {  
 res[i] = array[i];  
 }  
 return res;  
}  
  
void bubble\_sort(int\* array) {  
 int i, j, temp;  
  
 for (i = 0; i < n - 1; i++) {  
 for (j = 0; j < n - i - 1; j++) {  
 if (array[j] > array[j + 1]) {  
 temp = array[j];  
 array[j] = array[j + 1];  
 array[j + 1] = temp;  
 }  
 }  
 }  
}  
  
void insertion\_sort(int\* array) {  
 int i, j, key;  
  
 for (i = 1; i < n; i++) {  
 key = array[i];  
 j = i - 1;  
  
 while (j >= 0 && array[j] > key) {  
 array[j + 1] = array[j];  
 j--;  
 }  
 array[j + 1] = key;  
 }  
}  
  
void selection\_sort(int\* array) {  
 int i, j, min, temp;  
  
 for (i = 0; i < n - 1; i++) {  
 min = i;  
  
 for (j = i + 1; j < n; j++) {  
 if (array[j] < array[min]) {  
 min = j;  
 }  
 }  
 if (min != i) {  
 temp = array[min];  
 array[min] = array[i];  
 array[i] = temp;  
 }  
 }  
}  
  
  
void display\_array(int\* array) {  
 int i;  
 printf("\nFirst 100 elements of the array: ");  
  
 for (i = 0; i < 100; i++) {  
 printf("%d ", array[i]);  
 }  
 printf("\n");  
}  
  
void display\_time(char name[], clock\_t start, clock\_t end) {  
 printf("\nTime taken using %s: %.2lfms\n", name, (double)(end - start) / CLOCKS\_PER\_SEC \* 1000);  
}

**Output**:

Enter the range of random numbers to generate: 1 5001  
Array before sorting:  
  
First 100 elements of the array: 2507 4947 3535 2626 1657 493 2716 1924 255 4735 3817 1714 3879 890 3612 2943 2800 2839 4304 1694 322 3293 509 3014 1500 1766 4420 3849 978 4060 394 4247 4768 4691 1872 2187 183 4587 4873 1200 82 3688 3677 3960 4577 2287 2664 2375 888 2729 4068 1972 1020 4576 4985 2520 2104 4404 2130 3081 4225 2524 3090 3991 2213 4961 1939 3158 308 1810 120 1153 496 3796 874 834 1844 3538 3971 2731 1265 3037 4702 3048 3375 448 1329 1240 4851 4221 4320 4074 2506 3171 3826 4719 3130 1527 3638 4201   
Array after sorting using bubble sort:  
  
First 100 elements of the array: 1 2 2 3 3 3 4 4 4 5 5 5 6 6 6 6 6 7 7 7 7 8 8 9 10 11 11 12 12 13 13 14 14 14 14 14 14 15 15 17 17 17 17 18 20 20 21 21 21 21 21 22 23 23 24 24 24 25 25 26 26 26 26 27 27 27 27 27 27 28 29 30 30 30 30 30 31 31 31 32 32 32 32 32 33 34 34 36 37 37 37 38 38 39 39 39 40 41 41 41   
  
Time taken using bubble sort: 126.17ms  
Array after sorting using insertion sort:  
  
First 100 elements of the array: 1 2 2 3 3 3 4 4 4 5 5 5 6 6 6 6 6 7 7 7 7 8 8 9 10 11 11 12 12 13 13 14 14 14 14 14 14 15 15 17 17 17 17 18 20 20 21 21 21 21 21 22 23 23 24 24 24 25 25 26 26 26 26 27 27 27 27 27 27 28 29 30 30 30 30 30 31 31 31 32 32 32 32 32 33 34 34 36 37 37 37 38 38 39 39 39 40 41 41 41   
  
Time taken using insertion sort: 51.81ms  
Array after sorting using selection sort:  
  
First 100 elements of the array: 1 2 2 3 3 3 4 4 4 5 5 5 6 6 6 6 6 7 7 7 7 8 8 9 10 11 11 12 12 13 13 14 14 14 14 14 14 15 15 17 17 17 17 18 20 20 21 21 21 21 21 22 23 23 24 24 24 25 25 26 26 26 26 27 27 27 27 27 27 28 29 30 30 30 30 30 31 31 31 32 32 32 32 32 33 34 34 36 37 37 37 38 38 39 39 39 40 41 41 41   
  
Time taken using selection\_sort: 68.83ms