4.2 WAP to Sort a given set of elements using the Quick Sort method and determine the time required to Sort the elements. Repeat the experiment for different value of n. Plot a graph time taken v/s n. The elements can be generated randomly.

Program :-

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
Void swap(int *a, int *b){
    int t = *a;
    *a = *b;
    *b = +;
}

int partition(int arr[], int low, int high){
    int pivot = arr[high];
    int i = (low - 1);
```

```
for (int j = low; j <= high - 1; j++) {
    if (arr[j] < pivot) {
        i++;
        Swap (&arr[i], &arr[j]);
    }
}
Swap (&arr[i+1], &arr[high]);
return (i+1);
}

void quickSort (int arr[], int low,
                int high){
if (low < high){
  int pi = partition(arr, low, high);
  quicksort (arr, low, pi-1);
  quicksort(arr, pi+1, high);
}
}
int main() {
    int i, n;
    clock_t start, end;
```

```
double total_cputime;
print ("Enter Size : ");
Scanf ("%d", &n);
int a[n];
start = clock ();
for ( i=0 ; i<n; i++) {
        a [i ]= (rand ()%50000);
        printf ("%d \t", a [i]);
}
printf ("\n Elements generated random-
                    -ly...\n");
quickSort (a, 0, n-1);
printf ("\n Array after quick Sort:\n");
    for (i=0 ; i<n ; i++)
            print ("%d \t", a [i]);
    end = clock ();
printf("\n\n CPU Time Calculation:: ");
total_Cpu time = ((double ) (end - Start ));
printf("\n Total CPU time: %f ms", total_cpu
                                    -time );
```

```
total-cputime = ((double)(end - start)/
                                clocks_per_SEC);

printf("\n Total CPU time: %f S", total-cputime);

return 0;
}
```

Q5.2 WAP to implementation of fractional knapsack algorithm.

Program:-

```
#include<stdio.h>
Struct knap{
    int p, w;
    float r, x;
};
Void print (struct knap l[], int n){
    int i, j;
    printf("\n");
    for (i=0; i<n; i++)
        printf("|%.d %.d %.0.2f %.0.2f|\n",
            l[i].p, l[i]w, l[i].r, l[i].x);
}
```

```
float Knapsack (int n, int m){
    int i,j;
    float profit = 0.0;
    struct knap k[10], t;
    for (i=0; i<n; i++){
        printf("Enter price and weight of
                item %d: "; i+1);
        Scanf("%d %d", &k[i].p, &k[i].w);
        k[i].r = (float)(k[i].p)/(float)(k[i].w);
    }
    printf(k, n);
    printf("At first array look like\n");
    for (i=0; i<n-1; i++){
        for (j=i; j<n; j++){
            if(k[i].r < k[j].r){
                t = k[i];
                k[i] = k[j];
                k[j] = t;
            }
        }
    }
```

```
print f (K, n);
print f ("After sorting the array w.r.t
                                R \n");

for (i = 0; i<n; i++) {
    if (K[i].w < m) {
        K [i].x = 1;
        m -= K [i].w;
    }
else {
    K [i].x = (float)(m) /(float)(K [i].w);
        m = 0;
    }
    profit += (float)(K [i]. p) * K [i]. x;
}
print f(K, n);
print f ("finally the array looks like \n");
return profit;
}
```

```
int main(){

    int n, m;
    printf("Enter Total item :");
    Scanf("%d", &n);
    printf("Enter Size of KnapSack :");
    Scanf("%d", &m);
    printf("\n");
    printf("\n Maximum profit will be:
                %.0.2f \n", KnapSack(n,m));

return 0;

}
```