Page No.
Date

Name Rohan Siwach USN 1BM19CS132
Date 10th June 2021, 11:15 LABTEST-1 ADA

LAB PROGRAM-9

Sort a given set of N integer elements using quick sort technique & compute its time taken

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

int partition(int a[], int low, int high)
{
  int pivot = a[high];
  int i = (low-1);
  for(int j = low; j <= high-1; j++)
  { if(a[j] < pivot)
    { i++;
      swap(&a[i], &a[j]);
    }
  }
  swap(&a[i+1], &a[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);
  }
}
void printarray (int arr[], int size)
{ int i;
  for (i=0; i<size; i++)
      printf ("%d", arr[i]);
  printf ("\n");
}
int main
{ int arr[15000], size, i, j, ch, temp;
  clock_t start, end;
  while(1)
  { printf ("\n1 for entry of N value & array ele");
    printf ("\n2 to display time taken");
    printf ("\n3 to exit the prog");
    printf ("\n Enter your choice");

  scanf ("%d", &ch);
  { case 1: printf ("Enter the size\n");
            scanf ("%d", &size);
            printf ("Enter the element\n");
            scanf ("%d", &arr[i]);
            start = clock();
            quicksort (arr, 0, size-1);
            end = clock();
            printf ("sorted array:");
            printarray (arr, size);
```

```
printf("\n time taken is %.f
    to sort %.d nos is %.f sec\n",
    size, ((double)(end-start))/
        clocks_per_sec));
break;

case 2: size = 500;
  while (size<=14500)
    { for (i=0; i<size; i++)
      { arr[i] = size-1;
      }
      start = clock();
      quicksort (arr, 0, size-1);
    // Dummy loop to create delay
      for (j=0; j<=50000; j++)
        { temp = 38/600;
        }
      end = clock();
       printf("\n time taken is %.d nos
         is %.f secs\n", size, ((double)(
         end-start))/clocks_per_sec));
      size = size + 1000;
    }
  break;
  case 3  exit(0);
      break;
  }
  return 0;
}
```

modification

```
int kth smallest (int * arr, int low, int
high, int k){
    if (k>0 && k<=high-low+1) {
        int pos = partition (arr, low
        , high);
    if (pos-low==k-1){
        return arr[pos];
    }
    else if (pos-low>k-1){
        return kth smallest (arr, low
    pos-1, k);
    }
    return kth smallest (arr, pos+1, high
    , k-(pos+1));
    }
}
```