Digital Assignment – 3

Name: Hari Krishna Shah

VIT ID: 21BCS0167

LINK :
https://drive.google.com/drive/folders/16_V2VfMSeV_rJ68gXOWWIkCougfDkCjT?usp=sharing

Ques 1 and 2 combined:

1. Assume in the Regional Passport Office, a multitude of applicants arrive each day for passport renewal. A list is maintained in the database to store the renewed passports arranged in the increased order of passport ID. The list already would contain there cords renewed till the previous day. Apply Insertion sort technique to place the current day's records in the list.

Later the office personnel wish to sort the records based on the date of renewal so as to know the count of renewals done each day. Taking into consideration the fact that each record has several fields (around 25 fields), follow Selection sort logic to implement the same.

2.Implement above using quick sort,and merge sorting techniques.

Answer:

```
#include <stdio.h>
#include <malloc.h>

int partition (struct today_record *a, int start, int end);
void quick(struct today_record *a, int start, int end);
void merge(struct today_record *a, int beg, int mid, int end);
void mergeSort(struct today_record *a, int beg, int end);
void database_day_sort(struct database *a, int size);

struct database{
     int passport_id;
     int day;
     int month;
     int year;
};
```

```
struct today_record{
     int passport_id;
     int day;
     int month;
     int year;
};

int main(){
     struct database *d;
     struct today_record *r;
     static int r_size = 0, d_size = 0;
     int option, sort_choice;
     d = (struct database *) (malloc(d_size*sizeof(struct
database)));

     do{
          printf("\nWelcome to the passport-id sorting
algorithm.\n");
          printf("Please choose an option from below menu.\n \
          Enter 1 to display the total database \n \
          Enter 2 to add today's passort-id to database\n \
          Enter 3 to sort the database according to the renewal
day\n");
          printf("Enter your option here: ");
          scanf("%d", &option);
          printf("\n");

          switch(option){
               case -1:{
                    printf("The program exited
successfully.\n");
                    break;
               }
               case 1:{
                    if(d_size == 0){
                         printf("Database empty.\n");
                    }
                    else{
                         printf("The passports ids stored in
the database are given below: \n");
                         for(int i = 0; i<d_size; i++){
                              printf("S.N: %d    Passport_ID:
%d\n", i+1, d[i].passport_id);
                              printf("Renewed Data: %d-%d-%d",
```

```
d[i].day, d[i].month, d[i].year);
                                   printf("\n\n");
                              }
                         }

                         break;
                    }
                    case 2:{
                         free(r);
                         printf("Enter the size of the today
record: ");
                         scanf("%d", &r_size);
                         r = (struct today_record *)
(malloc(r_size*sizeof(struct today_record)));
                         d = (struct database *) (realloc(d,
(d_size+r_size)*sizeof(struct database)));

                         for(int i = 0; i<r_size; i++){
                              printf("Enter the id number %d: ",
i+1);
                              scanf("%d", &r[i].passport_id);
                              printf("Enter the renwal day: ");
                              scanf("%d", &r[i].day);
                              printf("Enter the renewal month: ");
                              scanf("%d", &r[i].month);
                              printf("Enter the renewal year: ");
                              scanf("%d", &r[i].year);
                              printf("\n\n");
                         }

                         // sorting
                         printf("The details for today's record has
accepted successfully.\n");
                         printf("Lets now sort the today's record
and then store it in the database.\n");
                         printf("Enter 1 to sort the record by
insertion sort.\n");
                         printf("Enter 2 to sort the record by
selection sort\n");
                         printf("Enter 3 to sort by quick
sort.\n");
                         printf("Enter 4 to sort by merger
sort.\n");
                         printf("Enter your option here: ");
                         scanf("%d", &sort_choice);
```

```
                    switch(sort_choice){
                         case 1:{
                              int i, key, j;
                             for (i = 1; i < r_size; i++) {
                                 key = r[i].passport_id;
                                 j = i - 1;
                                 while (j >= 0 &&
r[i].passport_id > key) {
                                     r[j + 1].passport_id =
r[j].passport_id;
                                     r[j + 1].day = r[j].day;
                                     r[j + 1].month =
r[j].month;
                                     r[j + 1].year =
r[j].year;

                                     j = j - 1;
                                 }
                                 r[j + 1].passport_id = key;
                             }

                              break;
                         }
                         case 2:{
                         struct today_record temp;
                         for(int i = 0; i<r_size; i++){
                              for(int j = i+1; j<r_size; j++){

     if(r[j].passport_id<r[i].passport_id){
                                         temp = r[i];
                                         r[i] = r[j];
                                         r[j] = temp;
                                   }
                              }
                         }
                         break;
                    }

                    case 3:{
                         quick(r, 0, r_size - 1);
                         break;
                    }

                    case 4:{
```

```
                    mergeSort(r, 0, r_size-1);
                    break;
                }


                }
                printf("\nThe array has been sorted
successfully.\n");
                //After sorting copying the sorted record
in the database
                for(int i = 0; i<r_size; i++){
                    d[d_size].passport_id =
r[i].passport_id;
                    d[d_size].day = r[i].day;
                    d[d_size].month = r[i].month;
                    d[d_size].year = r[i].year;
                    d_size += 1;
                }



                    break;
                }

                case 3:{
                    database_day_sort(d, d_size);
                    break;
                }


                printf("\n");
            }
        }
        while(option!= -1);
}


//quick sort
int partition (struct today_record *a, int start, int end)
{
    int pivot = a[end].passport_id; // pivot element
    int i = (start - 1);
    struct today_record temp;

    for (int j = start; j <= end - 1; j++)
```

```
    {
        // If current element is smaller than the pivot
        if (a[j].passport_id < pivot)
        {
            i++; // increment index of smaller element
            temp= a[i];
            a[i] = a[j];
            a[j] = temp;
        }
    }
    temp = a[i+1];
    a[i+1] = a[end];
    a[end] = temp;
    return (i + 1);
}

/* function to implement quick sort */
void quick(struct today_record *a, int start, int end) /* a[] =
array to be sorted, start = Starting index, end = Ending index
*/
{
    if (start < end)
    {
        int p = partition(a, start, end); //p is the
partitioning index
        quick(a, start, p - 1);
        quick(a, p + 1, end);
    }
}

//Merge Sort
void merge(struct today_record *a, int beg, int mid, int end)
{
    int i, j, k;
    int n1 = mid - beg + 1;
    int n2 = end - mid;

    struct today_record LeftArray[n1], RightArray[n2];
//temporary arrays

    /* copy data to temp arrays */
    for (int i = 0; i < n1; i++)
    LeftArray[i] = a[beg + i];
    for (int j = 0; j < n2; j++)
    RightArray[j] = a[mid + 1 + j];
```

```
    i = 0, /* initial index of first sub-array */
    j = 0; /* initial index of second sub-array */
    k = beg;  /* initial index of merged sub-array */

    while (i < n1 && j < n2)
    {
        if(LeftArray[i].passport_id <
RightArray[j].passport_id)
        {
            a[k] = LeftArray[i];
            i++;
        }
        else
        {
            a[k] = RightArray[j];
            j++;
        }
        k++;
    }
    while (i<n1)
    {
        a[k] = LeftArray[i];
        i++;
        k++;
    }

    while (j<n2)
    {
        a[k] = RightArray[j];
        j++;
        k++;
    }
}
void mergeSort(struct today_record*a, int beg, int end)
{
    if (beg < end)
    {
        int mid = (beg + end) / 2;
        mergeSort(a, beg, mid);
        mergeSort(a, mid + 1, end);
        merge(a, beg, mid, end);
    }
}
```

```c
//sorting according to the day

void database_day_sort(struct database *a, int size){
	struct database temp;
	for(int i = 0; i<size; i++){
		for(int j = i+1; j<size; j++){
			if(a[j].day < a[i].day){
				temp = a[i];
				a[i] = a[j];
				a[j] = temp;
			}
		}
	}
	printf("The database has been sorted successfully
according to the renewal day.\n");
}
```