**Digital Assignment – 3**

**Name: Hari Krishna Shah**

**VIT ID: 21BCS0167**

**LINK :** [**https://drive.google.com/drive/folders/16\_V2VfMSeV\_rJ68gXOWWIkCougfDkCjT?usp=sharing**](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**);**

**}**

**}**

*//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");**

**}**

