Q.1 WAP in C to simulate a car parking system consisting of 5 parking lanes each having a capacity to park 5 cars. Cars arrive from one end and depart from the other.

Display a menu having the following:

1. Read input from the file dynamically for each time slot. (to get the current status of parking system).

1. New Arrival of a car (to display the parking slot no. in the respective lane).

2. Departure of the first car in the lane.

3. Departure of the cars other than the first one(the ones parked behind in the lane). In this display the parking slot of the car where it is parked currently.

Make separate methods/functions for each choice.

------------------------------------------------------

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <wchar.h>

struct node {

int data; char reg[20]; struct node \*prev, \*next; };

struct node \*lane\_head[5],\*lane\_tail[5];

int lane\_count[5]={0}; int car\_count=0;

struct node \* createNode(int data,char \*reg) {

struct node \*newnode = (struct node \*)malloc(sizeof (struct node));

newnode->data = data;

char a[20];

strcpy(newnode->reg,reg);

newnode->next = newnode->prev = NULL;

return (newnode); }

/\*initialize every head and tail pointer for insertions\*/

void createSentinels(int lane) {

lane\_head[lane] = createNode(0,"");

lane\_tail[lane] = createNode(0,"");

lane\_head[lane]->next = lane\_tail[lane];

lane\_tail[lane]->prev = lane\_head[lane]; }

/\* insertion at the front of the queue \*/

void enqueueAtFront(int data,char \*reg,int lane) {

struct node \*newnode, \*temp;

newnode = createNode(data,reg);

temp = lane\_head[lane]->next;

lane\_head[lane]->next = newnode;

newnode->prev = lane\_head[lane];

newnode->next = temp;

temp->prev = newnode; }

/\*Initialize every lane for insertion and deletion\*/

void initialize(){

int i=0;

for(i=0;i<5;i++){ lane\_head[i]=NULL; lane\_tail[i]=NULL; createSentinels(i); }}

/\*Car insert at front\*/

void add\_car(int time,char reg[],int lane,int orig) {

if((car\_count<=25)) {

if((lane\_count[lane]<=5)){

enqueueAtFront(time,reg,lane);

if(orig==0) printf("Car parked at front of Lane: %d\n",lane\_count[lane]+1);

lane\_count[lane]++;

} else enqueueAtFront(time,reg,(lane+1)%5);

} else printf("!\*\*Total Capacity Full\*\*!\n"); }

/\* deletion at a special position\*/

void dequeueAtSpecial(char reg[]) {

int lane=search(reg,2);

if(lane>=0){ struct node \*temp;

if (lane\_head[lane]->next==lane\_tail[lane]) printf("Queue is empty\n");

else { temp = lane\_head[lane]->next;

while((temp!=lane\_tail[lane])) {

if(strcmp(temp->reg,reg)==0){

temp->prev->next = temp->next; temp->next->prev = temp->prev;

temp->next=NULL; temp->prev=NULL;

free(temp); car\_count--; lane\_count[lane]--;

break; }

temp = temp->next; } } } }

/\* check\_slots elements present in the queue \*/

void check\_slots(int lane) {

struct node \*temp; int count=0; printf("Lane\_%d: ",lane+1);

if (lane\_head[lane]->next == lane\_tail[lane]) {

printf("Lane\_%d is empty\n",lane); return; }

temp = lane\_tail[lane]->prev;

while(temp!=lane\_head[lane]) {

count++; temp= temp->prev; }

int slots\_left=5-count; printf("%d slots are available.\n",slots\_left); }

/\*search a car in every lane sequentialy and then return lane no\*/

int search(char reg[],int orig) {

struct node \*temp; int i,flag=0,lane,slot=0;

for(i=0;i<5;i++) { lane=i;

if (lane\_head[lane]->next == lane\_tail[lane]) return -1;

temp = lane\_tail[lane]->prev;

while(temp!=lane\_head[lane]) { slot++;

if(strcmp(reg,temp->reg)==0) {

if(orig==2){ printf("CAR: %s DEPARTED\nFrom Lane: %d ; Slot: %d which arrived at time %d \n",reg,i+1,temp->data,slot);

flag=1; break; }

else{ printf("%s Car is in Lane %d ; Arrived at time %d \n",reg,i+1,temp->data);

flag=1; break; } }

temp= temp->prev; }

if(flag==1) break; }

if(flag==0){ printf("no vehical found\n"); return -1; }

return lane; }

/\*check\_slots current parking lanes\*/

void show\_status() {

for(i=0;i<5;i++) {

int lane=i; struct node \*temp; printf("Lane\_%d: ",lane+1);

if (lane\_head[lane]->next == lane\_tail[lane]) {

printf("Lane\_%d is empty\n",lane); return; }

temp = lane\_tail[lane]->prev;

while(temp!=lane\_head[lane]) {

printf("\xF0\x9F\x9A\x90"); printf(" %s", temp->reg);

printf("\xF0\x9F\x92\xA8"); printf(" ");

temp= temp->prev; } printf("\n"); } }

int console() {

int choice;

printf("\n1. Enter(Continue) Simulator\n0. Exit Simulator\n");

scanf("%d",&choice);

if(choice==1) {

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("|\t\tEnter Your Choice |\n"); printf("|\t\t----------------- |\n");

printf("|\t |\n"); printf("|\t0. Exit |\n");

printf("|\t1. Show current status |\n");

printf("|\t2. Show parking slots available in each lane |\n");

printf("|\t3. Add a New Car |\n");

printf("|\t4. Departure of first Car |\n");

printf("|\t5. Departure of specific Car |\n");

printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n");

scanf("%d",&choice); } return choice; }

/\*remove car first search than remove car\*/

void remove\_car(int lane\_input) {

if(lane\_input>=0 && lane\_input<=5){

struct node \*temp;

if (lane\_head[lane\_input]->next==lane\_tail[lane\_input]) printf("Queue is empty\n");

else temp = lane\_head[lane\_input]->next;

while((temp!=lane\_tail[lane\_input])) {

if(1){

temp->prev->next = temp->next; temp->next->prev = temp->prev;

temp->next=NULL; temp->prev=NULL;

free(temp); car\_count--; lane\_count[lane\_input]--;

printf("FIRST Car IN LANE %d DEPARTED\n",lane\_input+1);

break;} temp = temp->next; } } }

else printf("Wrong Lane entered"); }

int main() {

FILE \*fp; char a[20],temp[20],reg[20]; int tim,time\_unit=0,I,orig=0;

initialize(); strcpy(a,"parking\_data.txt"); fp = fopen(a,"r");

if(fp) { orig=1;

do { fscanf(fp,"%s",reg); fscanf(fp,"%s",temp); tim=atoi(temp);

if(time\_unit==tim){ time\_unit=tim; car\_count++;

add\_car(tim,reg,(car\_count-1)%5,orig); }

else if(time\_unit<tim){ time\_unit=tim; car\_count++;

add\_car(tim,reg,(car\_count-1)%5,orig); }}while(!feof(fp)); }

int y,choice,lane\_input; printf("\n!!....Welcome....!!\n");

while(choice = console()){ switch(choice){

case 0: break;

case 1: show\_status(); break;

case 2: for(y=0;y<5;y++) check\_slots(y); break;

case 3: printf("Input Registration no: \n"); scanf("%s",reg);

printf("Input time:\n"); scanf("%d",&tim); orig=0;

if(time\_unit<=tim) { time\_unit=tim; car\_count++;

add\_car(tim,reg,(car\_count-1)%5,orig); }

else printf("Incorrect time\n"); break;

case 4: printf("Enter the Lane no: \n");

scanf("%d",&lane\_input);remove\_car(lane\_input-1); break;

case 5: printf("Enter the regestration no: \n");

scanf("%s",reg);dequeueAtSpecial(reg); break;

default: printf("wrong choice\n"); } }

}

!!....Welcome....!!

1. Enter(Continue) Simulator

0. Exit Simulator

1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| Enter Your Choice |

| ---------------------- |

| |

| 0. Exit |

| 1. Show current status |

| 2. Show parking slots available in each lane |

| 3. Add a New Car |

| 4. Departure of first Car |

| 5. Departure of specific Car |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

1

Lane\_1: 🚐 C1💨 🚐 C6💨 🚐 C11💨 🚐 C16💨 🚐 C21💨

Lane\_2: 🚐 C2💨 🚐 C7💨 🚐 C12💨 🚐 C17💨 🚐 C23💨

Lane\_3: 🚐 C3💨 🚐 C8💨 🚐 C13💨 🚐 C20💨 🚐 C22💨

Lane\_4: 🚐 C4💨 🚐 C9💨 🚐 C14💨 🚐 C19💨 🚐 C22💨

Lane\_5: 🚐 C5💨 🚐 C10💨 🚐 C15💨 🚐 C18💨

1. Enter(Continue) Simulator

0. Exit Simulator

1

5

Enter the regestration no:

C10

CAR: C10 DEPARTED

From Lane: 5 ; Slot: 5 which arrived at time 22

1. Enter(Continue) Simulator

0. Exit Simulator

1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| Enter Your Choice |

| ---------------------- |

| |

| 0. Exit |

| 1. Show current status |

| 2. Show parking slots available in each lane |

| 3. Add a New Car |

| 4. Departure of first Car |

| 5. Departure of specific Car |

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

1

Lane\_1: 🚐 C1💨 🚐 C6💨 🚐 C11💨 🚐 C16💨 🚐 C21💨

Lane\_2: 🚐 C2💨 🚐 C7💨 🚐 C12💨 🚐 C17💨 🚐 C23💨

Lane\_3: 🚐 C3💨 🚐 C8💨 🚐 C13💨 🚐 C20💨 🚐 C22💨

Lane\_4: 🚐 C4💨 🚐 C9💨 🚐 C14💨 🚐 C19💨 🚐 C22💨

Lane\_5: 🚐 C5💨 🚐 C15💨 🚐 C18💨

Q.2 Write a c program to count the number of characters, words, lines, in a text file (take input in both ways: as command line argument, from a file).

#include <stdio.h>

#include <string.h>

int main(){

printf("Enter Choice:\n");

printf("1. Input from File\n2. Input from Command Line ('$' to stop).\n3. Exit\n");

FILE \*f;

int choice, no\_line=0, no\_word=0, no\_character=0, size,i;

char filename[100],ch,para[1000];

scanf("%d",&choice);

switch(choice){

case 1: printf("Enter filename:"); scanf("%s", filename); f = fopen(filename,"r");

if(f){ while((ch=getc(f))!=EOF){

if (ch!=' ' && ch!='\n') ++no\_character;

if(ch==' ' || ch=='\n') ++no\_word;

if (ch=='\n') ++no\_line; }

if(no\_character>0){ ++no\_line; ++no\_word; } }

else{ printf("Unable to open file\n"); break; }

printf("Lines : %d \nWords : %d \nCharacters : %d \n", no\_line-1,no\_word-1,no\_character); fclose(f); break;

case 2: printf("Enter paragraph:\n"); scanf("%[^\$]",para); size=strlen(para)-1;

for(i=0;i<size;i++){ if (para[i]!=' ' && para[i]!='\n') ++no\_character;

if(para[i]==' ' || para[i]=='\n') ++no\_word;

if (para[i]=='\n') ++no\_line; }

if(no\_character>0){ ++no\_line; ++no\_word; }

printf("Lines : %d \nWords : %d \nCharacters : %d \n", no\_line-1,no\_word-1,no\_character+1); break;

case 3: break;

default: printf("Wrong Choice!\n"); break;

} return(0);

}

Q.3 Use the attached file "numbers.txt" and write a C program to do the given tasks:

1) Read the numbers from “numbers.txt” and write them into a new file named “numbers\_1.txt” in which each line has only 10 numbers.

2) Take a number from the user as input and search the file “numbers\_1.txt” to see if that number exists in the file. If yes, print the line number and position of it in that line on which it occurs. If not, display: “Number does not exists in the file.”

3) Replace the number searched previously by incrementing the old number by 1, in “numbers\_1.txt”.

4) Delete the whole line from the file “numbers\_1.txt” in which the replaced number occurs.

5) Sort the remaining numbers of “numbers\_1.txt” in ascending order using MERGE sort and write them into a new file “numbers\_2.txt”.

6) Compare the files “numbers\_1.txt” and “numbers\_2.txt” line by line and print the different lines.

#include<stdio.h>

#include <stdlib.h>

int X[10],Y[10],k=0,arr[100];

void merge(int arr[], int l, int m, int r);

void mergeSort(int arr[], int l, int r);

int main(){

char ch; int numbers\_1,i=0,c,x,flag=0,flag1=0;

FILE \*fp=fopen("numbers.txt","r"); FILE \*fptr=fopen("numbers\_1.txt","w");

while(fscanf(fp,"%d%c",&numbers\_1,&ch)!=EOF){

if(i==10){ i=0; fprintf(fptr, "\n"); }

fprintf(fptr, "%d ",numbers\_1); i++; }

printf("---------------------------\n Enter number to be searched: ---------------------------\n\t\t ");

scanf("%d",&x); printf("------------------------------\n");

i=0,c=1; fseek(fptr, 0, SEEK\_SET);

fptr=fopen("numbers\_1.txt","r+");

while(fscanf(fptr,"%d",&numbers\_1)!=EOF){

if(i==10) c++,i=0;

if(numbers\_1==x){

printf("\n\t\t|\n\t\t|\n\t\t|\n\t\t|\n\t\t|\n\t\tV\n------------------------------\n\tNumber found!! ");

printf("Line-> %d\t Position-> %d\n------------------------------\n\n\n ",c, i+1); flag=1; fseek(fptr, -2, SEEK\_CUR);

fprintf(fptr, "%d",x+1); X[k]=c; Y[k]=i+1; k++; }i++; }

if(!flag){ flag1=1;

printf("\n\n\n-----------------------------------------------\n Number does not exists in the file "); }

if(flag1!=1){ printf("\n\n\n---------------------------------------------------------------\n");

printf("You want to delete the line that contain the number %d (1/0)\n",x);

printf("---------------------------------------------------------------\n\t\t\t");

scanf("%d",&c); printf("---------------------------------------------------------------\n");

if(c==1){ for(int i=0;i<k;i++){

int lineNo=X[i]-i; FILE \*fileptr1, \*fileptr2; char ch; int temp = 1;

fileptr1 = fopen("numbers\_1.txt", "r"); fileptr2 = fopen("temp.txt", "w");

while((ch=getc(fileptr1))!=EOF){ if (ch=='\n') temp++;

if(temp!=lineNo) putc(ch,fileptr2); }

fclose(fileptr1); fclose(fileptr2); remove("numbers\_1.txt");

rename("temp.txt", "numbers\_1.txt"); }

printf("\n\n\n----------------------------\n !!\_\_Line Deleted\_\_!!\n ----------------------------\n "); }

else if(c==0)

printf("\n\n\n--------------------------\n!!\_\_Line not Deleted\_\_!!\n --------------------------\n "); }

else

printf("\nNumber not found so line could not be deleted!!\n---------------------------------\n ");}

printf("\n\n\n---------------------------------------------------------\n Do you want to sort the numbers (1/0):\n[It will create a seperate file named sortedNumber.txt]\n ---------------------------------------------------------\n\t\t\t ");

scanf("%d",&c); printf("---------------------------------------------------------\n");

if(c==1){ rewind(fptr); fptr=fopen("numbers\_1.txt","r");

k=0; while(fscanf(fptr,"%d",&arr[k++])!=EOF);

mergeSort(arr, 0, k - 1); fp=fopen("sortedNumber.txt","w");

rewind(fp); c=0;

while(c<k){ if(i==10){ fprintf(fp, "\n"); }

fprintf(fp, "%d ",arr[c]); i++; c++; }

printf("\n\n\n-----------------------------\n!!\_\_File has been soted\_\_!!\n ------------------\n "); }

else if(c==0)

printf("\n\n\n-----------------------------\n!!\_\_File not soted\_\_!!\n -----------------------------\n ");

}

void merge(int arr[], int l, int m, int r){

int i=0, 0, k=1,n1 = m - l + 1,n2 = r – m,L[n1], R[n2];

for (i = 0; i < n1; i++) L[i] = arr[l + i];

for (j = 0; j < n2; j++) R[j] = arr[m + 1+ j];

while(i < n1 && j < n2){

if (L[i] <= R[j]) { arr[k] = L[i]; i++; }

else { arr[k] = R[j]; j++; } k++; }

while (i < n1) { arr[k] = L[i]; i++; k++; }

while (j < n2) { arr[k] = R[j]; j++; k++; }

}

void mergeSort(int arr[], int l, int r){

if (l < r){ int m = l+(r-l)/2; mergeSort(arr, l, m);

mergeSort(arr, m+1, r); merge(arr, l, m, r); }

}

Q.4 Write a c program to count and display the operators, valid identifiers and keywords in the input text (Take input as command line argument).

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stddef.h>

void get\_keywords(char string[]);

void get\_operators(char string[]);

int main() {

printf("Enter the line of code:"); char string[100];

scanf("%[^\n]s", string); printf("%s\n", string);

printf("Operartors: \n") ; get\_operators(string);

printf("Keywords: \n") ; get\_keywords(string); }

void get\_keywords(char string[]){

int c=10,i=0,j=0,i\_count=0;

char keyword[15][20]= {"int","long","main","scanf","printf","char","short","string","do","while","for","float","double","break","continue"};

char \*identifier[100];

for(i=0;i<c;i++) {

int count=0,flag; char \*token,\*str = strdup(string);

while ((token = strsep(&str, " ,:=;.\*/+-()"))) {

if(strcmp(keyword[i],token)==0) count++;

else if(strcmp("",token)!=0) { flag=0;

for(j=0;j<c;j++) { if(strcmp(keyword[j],token)==0) { flag=1; break; } }

for(j=0;j<i\_count;j++) {

if(strcmp(identifier[j],token)==0) { flag=1; break; } }

if(flag==0) {

if((int)token[0]<48||(int)token[0]>57){ identifier[i\_count] = &token[0]; i\_count++; }

} } }

if(count!=0) printf("%s\t:\t%d \n",keyword[i],count); }

printf("Identifier: \n") ;

for(i=0;i<i\_count;i++) {

int count=0,flag;

char \*token ,\*str = strdup(string);

while ((token = strsep(&str, " ,=:;.\*/+-()"))) {

if(strcmp(identifier[i],token)==0) count++; }

if(count!=0) printf("%s\t:\t%d \n",identifier[i],count); } }

void get\_operators(char string[]) {

char operators[]=",:;=.\*/+-()#", \*str = strdup(string);

int i=0;

for(i=0;i<strlen(operators);i++) {

char \*token; str = strdup(string); char a[2]= "\0"; a[0] = operators[i];

int count=0; while ((token = strsep(&str, a))) count++;

if(count-1!=0) printf("%s\t:\t%d \n",a,count-1); }

free(str); }