“ASSIGNMENT No. 1”

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");               }       }
```

}

OUTPUT-

!!....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=3 🚗 C6=3 🚗 C11=3 🚗 C16=3 🚗 C21=3

Lane_2: 🚗 C2=3 🚗 C7=3 🚗 C12=3 🚗 C17=3 🚗 C23=3

Lane_3: 🚗 C3=3 🚗 C8=3 🚗 C13=3 🚗 C20=3 🚗 C22=3

Lane_4: 🚗 C4=3 🚗 C9=3 🚗 C14=3 🚗 C19=3 🚗 C22=3

Lane_5: 🚗 C5=3 🚗 C10=3 🚗 C15=3 🚗 C18=3

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=3 🚗 C6=3 🚗 C11=3 🚗 C16=3 🚗 C21=3

Lane_2: 🚗 C2=3 🚗 C7=3 🚗 C12=3 🚗 C17=3 🚗 C23=3

Lane_3: 🚗 C3=3 🚗 C8=3 🚗 C13=3 🚗 C20=3 🚗 C22=3

Lane_4: 🚗 C4=3 🚗 C9=3 🚗 C14=3 🚗 C19=3 🚗 C22=3

Lane_5: 🚗 C5=3 🚗 C15=3 🚗 C18=3

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);
}
```

OUTPUT-

```
Enter Choice:
1. Input from File
2. Input from Command Line ('$' to stop).
3. Exit
2
Enter paragraph:
This is a paragraph
which has eleven words
and three lines$
Lines : 3
Words : 11
Characters : 48

Enter Choice:
1. Input from File
2. Input from Command Line ('$' to stop).
3. Exit
1
Enter filename: test.txt
Lines : 3
Words : 13
Characters : 48
```

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);            }
}
```

OUTPUT-

```
-------------------------------------
Enter number to be searched:
-------------------------------------
                    384
-------------------------------------
                |
                |
                |
                |
                |
               V
-----------------------------------
          Number found!!
Line-> 1          Position-> 1
-----------------------------------
```

You want to delete the line that contain the number 384 (1/0)

1

!!__Line Deleted__!!

Do you want to sort the numbers (1/0):

[It will create a seperate file named sortedNumber.txt]

1

!!__File has been soted__!!

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","brea
k","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);                              }
```

Output-

Enter the line of code:int a,b,c;

int a,b,c;

Operartors:

, : 2

; : 1

Keywords:

int : 1

Identifier:

a : 1

b : 1

c : 1