SIDDAGANGA INSTITUTE OF TECHNOLOGY

www.sit.ac.in

Data Structures Laboratory

LAB MANUAL

Prabodh C P
Asst Professor
Dept of CSE, SIT

Contents

Data Structures Laboratory

Instructions

- All the C programs need to be executed using GCC Compiler.
- Algorithms and Flowcharts are compulsory for all the programs.
- All experiments must be included in practical examinations.

References

Part A: Behrouz A. Forouzan , Richard F. Gilberg , Computer Science: **A Structured programming Approach Using C** - Cengage Learning; 3rd edition
For writing flowcharts refer to **Appendix C** of the above book.

Chapter 1

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_ID/SALARY/DEPARTMENT/AGE. In case if the required record is not found, suitable message should be displayed.**

C Code - Text I/O

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct{
    unsigned emp_id;
    char emp_name[25];
    char emp_dept[25];
    unsigned emp_salary, emp_age;
}employee_t;

void fnAddRecord(void);
void fnSearchEmpID(int);
void fnSearchEmpSal(int);
void fnSearchEmpDept(char[]);
void fnSearchEmpAge(int);
void fnDisplayAllRecords(void);

int main()
{
    int id, sal, age, iChoice;
        char dept[10];

        for(;;)
        {
            printf("\n1.Add Record\n2.Display Records\n3.Search Employee by ID\n");
            printf("4.Search Employee by Dept\n5.Search Employee by salary\n");
            printf("6.Search Employee by Age\n7.Exit");
            printf("\nEnter your choice : ");
            scanf("%d",&iChoice);

            switch(iChoice)
```

```
            {
            case 1:
                    fnAddRecord();
                    break;

            case 2:
                    printf("\n Employee Details \n");
                    fnDisplayAllRecords();
                    break;

            case 3:
                    printf("\nEnter the emp_id that you want to search\n");
                    scanf("%d",&id);
                    fnSearchEmpID(id);
                    break;

            case 4:
                    printf("\nEnter the dept that you want to search\n");
                    scanf("%s",dept);
                    fnSearchEmpDept(dept);
                    break;

            case 5:
                    printf("\nEnter the salary that you want to search\n");
                    scanf("%d",&sal);
                    fnSearchEmpSal(sal);
                    break;

            case 6:
                    printf("\nEnter the age that you want to search\n");
                    scanf("%d",&age);
                    fnSearchEmpAge(age);
                    break;
            case 7: exit(0);
            }
        }
    return 0;
}

void fnDisplayAllRecords()
{
        int iCount = 0;
        employee_t ep;
        FILE *fp;

        fp = fopen("emp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fscanf(fp,"%d%s%s%d%d",&ep.emp_id, ep.emp_name, ep.emp_dept,
         &ep.emp_salary, &ep.emp_age)!=EOF)
        {
                printf("%d\t%s\t%s\t%d\t%d\n",ep.emp_id, ep.emp_name, ep.emp_dept, ep.emp_salary, ep.emp
                iCount++;
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
        fclose(fp);
}
```

```
void fnAddRecord()
{
        FILE *fp;
        employee_t emp;

    printf("\nEnter Employee details\n");
    printf("\nID : ");
    scanf("%d",&emp.emp_id);     getchar();
    printf("\nName : ");
    fgets(emp.emp_name,25,stdin);
    printf("\nDept : ");
    fgets(emp.emp_dept,25,stdin);
    printf("\nSalary : ");
    scanf("%d",&emp.emp_salary);
    printf("\nAge : ");
    scanf("%d",&emp.emp_age);

        fp = fopen("emp.dat", "a");
        fprintf(fp,"%d\t%s\t%s\t%d\t%d\n",emp.emp_id, emp.emp_name, emp.emp_dept, emp.emp_salary, emp.em
        fclose(fp);
}

void fnSearchEmpID(int id)
{
        int iCount = 0;
        employee_t ep;
        FILE *fp;

        fp = fopen("emp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fscanf(fp,"%d%s%s%d%d",&ep.emp_id, ep.emp_name, ep.emp_dept, &ep.emp_salary, &ep.emp_age)!
        {
                if(ep.emp_id == id)
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",ep.emp_id, ep.emp_name, ep.emp_dept, ep.emp_salary
                        iCount++;
                }
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
        fclose(fp);
}

void fnSearchEmpSal(int sal)
{
        int iCount = 0;
        employee_t ep;
        FILE *fp;

        fp = fopen("emp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fscanf(fp,"%d%s%s%d%d",&ep.emp_id, ep.emp_name, ep.emp_dept, &ep.emp_salary, &ep.emp_age)!
        {
                if(ep.emp_salary == sal)
```

```
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",ep.emp_id, ep.emp_name, ep.emp_dept, ep.emp_salary
                        iCount++;
                }
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
        fclose(fp);
}

void fnSearchEmpDept(char dept[])
{
        int iCount = 0;
        employee_t ep;
        FILE *fp;


        fp = fopen("emp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fscanf(fp,"%d%s%s%d%d",&ep.emp_id, ep.emp_name, ep.emp_dept, &ep.emp_salary, &ep.emp_age)!
        {
                if(!strcmp(ep.emp_dept, dept))
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",ep.emp_id, ep.emp_name, ep.emp_dept, ep.emp_salary
                        iCount++;
                }
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
}

void fnSearchEmpAge(int age)
{
        int iCount = 0;
        employee_t ep;
        FILE *fp;

        fp = fopen("emp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fscanf(fp,"%d%s%s%d%d",&ep.emp_id, ep.emp_name, ep.emp_dept, &ep.emp_salary, &ep.emp_age)!
        {
                if(ep.emp_age == age)
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",ep.emp_id, ep.emp_name, ep.emp_dept, ep.emp_salary
                        iCount++;
                }
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
}
```

C Code - Binary I/O

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct{
    unsigned emp_id;
    char emp_name[25];
    char emp_dept[25];
    unsigned emp_salary, emp_age;
}employee_t;

void fnAddRecord(void);
void fnSearchEmpID(int);
void fnSearchEmpSal(int);
void fnSearchEmpDept(char[]);
void fnSearchEmpAge(int);
void fnDisplayAllRecords(void);

int main()
{
    int id, sal, age, iChoice;
        char dept[10];
        printf("%lu bytes\n",sizeof(employee_t));
        for(;;)
        {
            printf("\n1.Add Record\n2.Display Records\n3.Search Employee by ID\n");
            printf("4.Search Employee by Dept\n5.Search Employee by salary\n");
            printf("6.Search Employee by Age\n7.Exit");
            printf("\nEnter your choice : ");
            scanf("%d",&iChoice);

            switch(iChoice)
            {
            case 1:
                    fnAddRecord();
                    break;

            case 2:
                    printf("\n Employee Details \n");
                    fnDisplayAllRecords();
                    break;

            case 3:
                    printf("\nEnter the emp_id that you want to search\n");
                    scanf("%d",&id);
                    fnSearchEmpID(id);
                    break;

            case 4:
                    printf("\nEnter the dept that you want to search\n");
                    scanf("%s",dept);
                    fnSearchEmpDept(dept);
                    break;

            case 5:
                    printf("\nEnter the salary that you want to search\n");
                    scanf("%d",&sal);
                    fnSearchEmpSal(sal);
                    break;

            case 6:
```

```
                    printf("\nEnter the age that you want to search\n");
                    scanf("%d",&age);
                    fnSearchEmpAge(age);
                    break;
            case 7: exit(0);
            }
        }
    return 0;
}

void fnDisplayAllRecords()
{
   int iCount = 0;
        employee_t rEmp;
        FILE *fp;

        fp = fopen("bemp.dat", "rb");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }

        while(fread(&rEmp, sizeof(employee_t),1,fp))
        {
                printf("%6d\t%15s\t%8s\t%8d\t%4d\n",rEmp.emp_id, rEmp.emp_name,
                                        rEmp.emp_dept, rEmp.emp_salary, rEmp.emp_age);
                iCount++;
                if(feof(fp))
                        break;
        }

        if(0 == iCount)
                printf("\nNo Records found\n");
        fclose(fp);
}

void fnAddRecord()
{
    FILE *fp;
    employee_t wEmp;

    printf("\nEnter Employee details\n");
    printf("\nID : ");
    scanf("%d",&wEmp.emp_id);                getchar();
    printf("\nName : ");
    gets(wEmp.emp_name);
    //fgets(wEmp.emp_name, 25, stdin);
    printf("\nDept : ");
    gets(wEmp.emp_dept);
    //fgets(wEmp.emp_dept, 25, stdin);
    printf("\nSalary : ");
    scanf("%d",&wEmp.emp_salary);
    printf("\nAge : ");
    scanf("%d",&wEmp.emp_age);

    fp = fopen("bemp.dat", "ab");

    fwrite(&wEmp, sizeof(employee_t),1,fp);
    //write(fp,&wEmp,sizeof(employee_t));

    fclose(fp);
```

```
}

void fnSearchEmpID(int id)
{
        int iCount = 0;
        employee_t sEmp;
        FILE *fp;

        fp = fopen("bemp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fread(&sEmp, sizeof(employee_t),1,fp))
        {
                if(sEmp.emp_id == id)
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",sEmp.emp_id, sEmp.emp_name,
                                sEmp.emp_dept, sEmp.emp_salary, sEmp.emp_age);
                        iCount++;
                }
                if(feof(fp))
                        break;
        }

        if(0 == iCount)
                printf("\nNo Records found\n");
        fclose(fp);
}

void fnSearchEmpSal(int sal)
{
        int iCount = 0;
        employee_t sEmp;
        FILE *fp;

        fp = fopen("bemp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fread(&sEmp, sizeof(employee_t),1,fp))
        {
                if(sEmp.emp_salary == sal)
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",sEmp.emp_id, sEmp.emp_name,
                                sEmp.emp_dept, sEmp.emp_salary, sEmp.emp_age);
                        iCount++;
                }
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
        fclose(fp);
}

void fnSearchEmpDept(char dept[])
{
        int iCount = 0;
        employee_t sEmp;
        FILE *fp;
```

```
        fp = fopen("bemp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fread(&sEmp, sizeof(employee_t),1,fp))
        {
                if(!strcmp(sEmp.emp_dept, dept))
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",sEmp.emp_id, sEmp.emp_name,
                                sEmp.emp_dept, sEmp.emp_salary, sEmp.emp_age);
                        iCount++;
                }
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
}

void fnSearchEmpAge(int age)
{
        int iCount = 0;
        employee_t sEmp;
        FILE *fp;

        fp = fopen("bemp.dat", "r");
        if(fp==NULL)
        {
            printf("\nFile does not exist\n");
            return;
        }
        while(fread(&sEmp, sizeof(employee_t),1,fp))
        {
                if(sEmp.emp_age == age)
                {
                        printf("%d\t%s\t%s\t%d\t%d\n",sEmp.emp_id, sEmp.emp_name,
                                sEmp.emp_dept, sEmp.emp_salary, sEmp.emp_age);
                        iCount++;
                }
        }
        if(0 == iCount)
                printf("\nNo Records found\n");
}
```

Output

Chapter 2

Stack Implementation

Question

Write a C program to implement STACK to perform the PUSH, POP and DISPLAY operations.

C Code Array Implementation

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define MAX 5

bool fnStkFull(int);
bool fnStkEmpty(int);
void fnPush(int [], int*);
int fnPop(int [], int*);
void fnDisplay(int[], int);
int fnPeek(int [], int);

int main()
{
        int stkArray[MAX];
        int top = -1;
        int iElem, iChoice;

        for(;;)
        {
                printf("\nSTACK OPERATIONS\n");
                printf("====================");
                printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.PEEK\n5.EXIT\n");
                printf("Enter your choice\n");
                scanf("%d",&iChoice);
                switch(iChoice)
                {
                        case 1: fnPush(stkArray, &top);
                                        break;

                        case 2: iElem = fnPop(stkArray, &top);
                                        if(iElem != -1)
                                                printf("\nPopped Element is %d\n", iElem);
                                        break;

                        case 3: fnDisplay(stkArray, top);
                                        break;

                        case 4: if(!fnStkEmpty(top))
```

```
                                        {
                                                iElem = fnPeek(stkArray, top);
                                                printf("\nElement at the  top of the stack is %d\n", iEl
                                        }
                                        else
                                                printf("\nEmpty Stack\n");
                                        break;

                        case 5: exit(1);

                        default: printf("\nWrong choice\n");
                }
        }
        return 0;
}

bool fnStkFull(int t)
{
        return ((t == MAX-1) ? true : false);
}

bool fnStkEmpty(int t)
{
        return ((t == -1) ? true : false);
}

void fnPush(int stk[], int *t)
{
        int iElem;
        if(fnStkFull(*t))
        {
                printf("\nStack Overflow\n");
                return;
        }
        printf("\nEnter element to be pushed onto the stack\n");
        scanf("%d", &iElem);

        *t = *t + 1;
        stk[*t] = iElem;
}

int fnPop(int stk[], int *t)
{
        int iElem;
        if(fnStkEmpty(*t))
        {
                printf("\nStack Underflow\n");
                return -1;
        }
        iElem = stk[*t];
        *t = *t - 1;

        return iElem;
}

void fnDisplay(int stk[], int t)
{
        int i;
        if(fnStkEmpty(t))
        {
                printf("\nStack Empty\n");
                return;
```

```
        }
        printf("\nStack Contents are: \n");
        for(i = t ; i > -1; --i)
        {
                printf("\t%d\n", stk[i]);
        }
}

int fnPeek(int stk[], int t)
{
        return stk[t];
}
```

C Code Structure Implementation

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define MAX 5

typedef struct{
        int stkArray[MAX];
        int top;
}STACK_TYPE;

bool fnStkFull(STACK_TYPE);
bool fnStkEmpty(STACK_TYPE);
void fnPush(STACK_TYPE*, int);
int fnPop(STACK_TYPE*);
void fnDisplay(STACK_TYPE);
int fnPeek(STACK_TYPE);

int main()
{

        STACK_TYPE myStack;
        myStack.top = -1;

        int iElem, iChoice;

        for(;;)
        {
                printf("\nSTACK OPERATIONS\n");
                printf("====================");
                printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.PEEK\n5.EXIT\n");
                printf("Enter your choice\n");
                scanf("%d",&iChoice);
                switch(iChoice)
                {
                        case 1: fnPush(stkArray, &top);
                                        break;

                        case 2: iElem = fnPop(stkArray, &top);
                                        if(iElem != -1)
                                                printf("\nPopped Element is %d\n", iElem);
                                        break;

                        case 3: fnDisplay(stkArray, top);
                                        break;
```

```
                        case 4: if(!fnStkEmpty(top))
                                        {
                                                iElem = fnPeek(stkArray, top);
                                                printf("\nElement at the  top of the stack is %d\n", iEl
                                        }
                                        else
                                                printf("\nEmpty Stack\n");
                                        break;

                        case 5: exit(1);

                        default: printf("\nWrong choice\n");
                }
        }
        return 0;
}

bool fnStkFull(int t)
{
        return ((t == MAX-1) ? true : false);
}

bool fnStkEmpty(int t)
{
        return ((t == -1) ? true : false);
}

void fnPush(int stk[], int *t)
{
        int iElem;
        if(fnStkFull(*t))
        {
                printf("\nStack Overflow\n");
                return;
        }
        printf("\nEnter element to be pushed onto the stack\n");
        scanf("%d", &iElem);

        *t = *t + 1;
        stk[*t] = iElem;
}

int fnPop(int stk[], int *t)
{
        int iElem;
        if(fnStkEmpty(*t))
        {
                printf("\nStack Underflow\n");
                return -1;
        }
        iElem = stk[*t];
        *t = *t - 1;

        return iElem;
}

void fnDisplay(int stk[], int t)
{
        int i;
        if(fnStkEmpty(t))
        {
```

```
                printf("\nStack Empty\n");
                return;
        }
        printf("\nStack Contents are: \n");
        for(i = t ; i > -1; --i)
        {
                printf("\t%d\n", stk[i]);
        }
}

int fnPeek(int stk[], int t)
{
        return stk[t];
}
```

Output

Chapter 3

Infix to Postfix Conversion

Question

Write a C program to convert the given infix expression to postfix expression.

C Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define STK_SIZE 10

void fnPush(char [], int*, char);
char fnPop(char [], int*);
int fnPrecd(char);

int main()
{
        int i, j=0;
        char acExpr[50], acStack[50], acPost[50], cSymb;
        int top = -1;

        printf("\nEnter a valid infix expression\n");
        scanf("%s", acExpr);

        fnPush(acStack, &top, '#');
        for(i=0;acExpr[i]!='\0'; ++i)
        {
                cSymb = acExpr[i];
                if(isdigit(cSymb))
                {
                        fnPush(acStack, &top, cSymb);
                }
                else if(cSymb == '(')
                {
                        fnPush(acStack, &top, cSymb);
                }
                else if(cSymb == ')')
                {
                        while(acStack[top] != '(')
                        {
                                acPost[j++] = fnPop(acStack, &top);
                        }
                        fnPop(acStack, &top);
                }
                else
```

```
                {
                        while(fnPrecd(acStack[top]) >= fnPrecd(cSymb))
                        {
                                acPost[j++] = fnPop(acStack, &top);
                        }
                        fnPush(acStack, &top, cSymb);
                }

        }
        while(acStack[top] != '#')
        {
                acPost[j++] = fnPop(acStack, &top);
        }
        acPost[j] = '\0';

        printf("\nInfix Expression is %s\n", acExpr);
        printf("\nPostfix Expression is %s\n", acPost);
        return 0;
}

void fnPush(char Stack[], int *t , char elem)
{
        *t = *t + 1;
        Stack[*t] = elem;

}

char fnPop(char Stack[], int *t)
{
        char elem;
        elem = Stack[*t];
        *t = *t -1;
        return elem;
}

int fnPrecd(char ch)
{
        switch(ch)
        {
                case '#' :          return -1;
                case '(' :          return 0;
                case '+' :
                case '-' :          return 1;
                case '*' :
                case '/' :          return 2;
        }
}
```

Output

Chapter 4

Evaluation of Prefix Expression

Question

Write a C program to evaluate the given prefix expression.

C Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define STK_SIZE 10

void fnPush(int [], int*, int);
int fnPop(int [], int*);

int main()
{
        int iaStack[50], i, iOp1, iOp2, iRes;
        char acExpr[50], cSymb;
        int top = -1;

        printf("\nEnter a valid prefix expression\n");
        scanf("%s", acExpr);

        for(i=strlen(acExpr)-1; i>=0; i--)
        {
                cSymb = acExpr[i];
                if(isdigit(cSymb))
                {
                        fnPush(iaStack, &top, cSymb-'0');
                }
                else
                {
                        iOp1 = fnPop(iaStack, &top);
                        iOp2 = fnPop(iaStack, &top);
                        switch(cSymb)
                        {
                                case '+' :      iRes = iOp1 + iOp2;
                                                break;
                                case '-' :      iRes = iOp1 - iOp2;
                                                break;
                                case '*' :      iRes = iOp1 * iOp2;
                                                break;
                                case '/' :      iRes = iOp1 / iOp2;
                                                break;
                        }
```

```
                        fnPush(iaStack, &top, iRes);
                }

        }
        iRes = fnPop(iaStack, &top);
        printf("\nValue of %s expression is %d\n", acExpr, iRes);
        return 0;
}

void fnPush(int Stack[], int *t , int elem)
{
        *t = *t + 1;
        Stack[*t] = elem;

}

int fnPop(int Stack[], int *t)
{
        int elem;
        elem = Stack[*t];
        *t = *t - 1;
        return elem;
}
```

Output

Chapter 5

Linear Queue Operations

Question

Write a C program to implement ordinary QUEUE to perform the insertion, deletion and display operations.

C Code

LinearQueue.c

```
/*****************************************************************************
*File                : LinearQueue
*Description        : Program to implement a Linear Queue of intrgers
*Author                : Prabodh C P
*Compiler        : gcc compiler4.4.3, Ubuntu 10.04
*Date                : 11 July 2012
*****************************************************************************/
#include<stdio.h>                                                              /*CPP*/
#include<stdlib.h>
#include "queue.h"
#include "QFunc.c"

/*****************************************************************************
*Function        :         main
*Input parameters:        no parameters
*RETURNS        :        0 on success
*****************************************************************************/
int main(void)
{
    QUEUE stQueue;
        int iChoice;
        stQueue.iFront = 0;
        stQueue.iRear = -1;
        for(;;)
        {
                printf("\nQueue Operations\n");
                printf("====================");

                printf("\n1.Qinsert\n2.Qdelete\n3.Qdisplay\n4.Exit\n");
                printf("Enter your choice\n");
                scanf("%d",&iChoice);
                switch(iChoice)
                {
                        case 1: stQueue = fnQInsert(stQueue);

                                break;
                        case 2: stQueue = fnQDelete(stQueue);
```

```
                                break;
                        case 3: fnQDisplay(stQueue);

                                break;
                        case 4: exit(0);
                        default: printf("\nWrong Choice\n");
                                break;
                }
        }
        return 0;
}
```

queue.h

```
#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED
#define SIZE 5
typedef struct
{
        int iaItems[SIZE];
        int iFront;
        int iRear;
}QUEUE;

QUEUE fnQInsert(QUEUE stQueue);
QUEUE fnQDelete(QUEUE stQueue);
void fnQDisplay(QUEUE stQueue);
int fnQFull(QUEUE stQueue);
int fnQEmpty(QUEUE stQueue);

#endif // QUEUE_H_INCLUDED
```

QFunc.c

```
/*****************************************************************************
*Function        :         fnQInsert
*Description:   inserts an element at the rear of the queue
*Input parameters: a structure queue
*RETURNS        :        updated queue
*****************************************************************************/

QUEUE fnQInsert(QUEUE stQueue)
{
        int iItem;
        if(fnQFull(stQueue))
        printf("\nQueue Overflow\n");
        else
        {
                printf("\nEnter the  element\n");
                scanf("%d",&iItem);
                stQueue.iRear++;
                stQueue.iaItems[stQueue.iRear] = iItem;

        }
        return stQueue;
}

/*****************************************************************************
*Function        :         fnQDelete
*Description:   deletes an element from the front of the queue
*Input parameters: a structure queue
*RETURNS        :        updated queue
*****************************************************************************/
```

```
QUEUE fnQDelete(QUEUE stQueue)
{
        if(fnQEmpty(stQueue))
        printf("\nQueue Underflow\n");
        else
        if(stQueue.iRear == stQueue.iFront)
        {
                printf("\nItem deleted is %d\n",stQueue.iaItems[stQueue.iFront]);
                stQueue.iRear=-1;
                stQueue.iFront=0;
        }
        else
        {
                printf("\nItem deleted is %d\n",stQueue.iaItems[stQueue.iFront++]);
        }
        return stQueue;
}

/*************************************************************************
*Function        :         fnQDisplay
*Description:   displays elements of the queue
*Input parameters: a structure queue
*RETURNS        :        nothing
*************************************************************************/
void fnQDisplay(QUEUE stQueue)
{
        int i;
        if(fnQEmpty(stQueue))
        printf("\nQueue Empty\n");
        else
        {
                printf("\nContents of Queue are:\n");
                for(i=stQueue.iFront;i<=stQueue.iRear;i++)
                printf("%d\t",stQueue.iaItems[i]);
        }
}
/*************************************************************************
*Function        :         fnQFull
*Description:   checks wheteher the queue is full or not
*Input parameters: a structure queue
*RETURNS        :        1 if the queue is full or 0 otherwise
*************************************************************************/
int fnQFull(QUEUE stQueue)
{
        if(stQueue.iRear == SIZE-1)
                return 1;
        else
                return 0;
}

/*************************************************************************
*Function        :         fnQEmpty
*Description:   checks wheteher the queue is empty or not
*Input parameters: a structure queue
*RETURNS        :        1 if the queue is empty or 0 otherwise
*************************************************************************/
int fnQEmpty(QUEUE stQueue)
{
        if(stQueue.iRear == stQueue.iFront-1)
                return 1;
        else
```

```
                return 0;
}
```

Output

Chapter 6

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output

Chapter 7

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output

Chapter 8

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output

Chapter 9

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output

Chapter 10

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output

Chapter 11

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output

Chapter 12

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output

Chapter 13

File Operations

Question

Write a C program to create a sequential file with at least five records, each record having the structure shown below:

EMPLOYEE_ID	NAME	DEPARTMENT	SALARY	AGE
Non-Zero +ve Integer	25 Characters	25 Characters	+ve Integer	+ve Integer

Write necessary functions to perform the following operations:

1. **to display all the records in the file.**
2. **to search for a specific record based on EMPLOYEE_IDSALARYDEPARTMENTAGE. In case if the required record is not found, suitable message should be displayed.**

C Code

Output