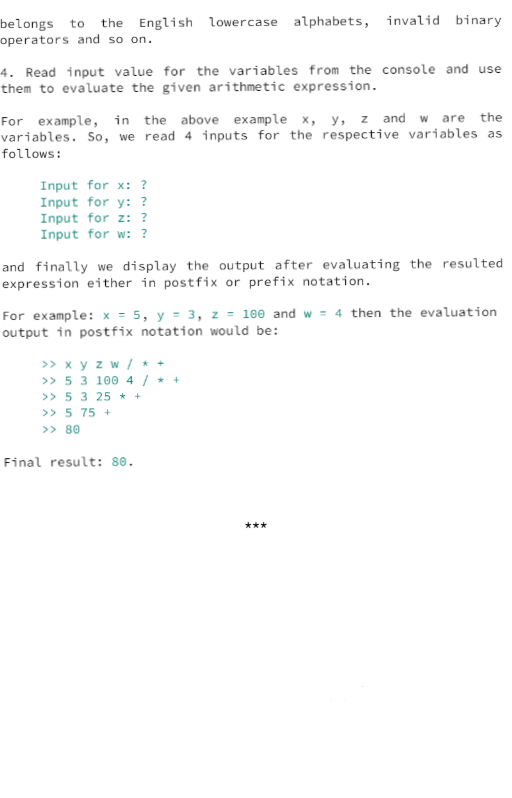
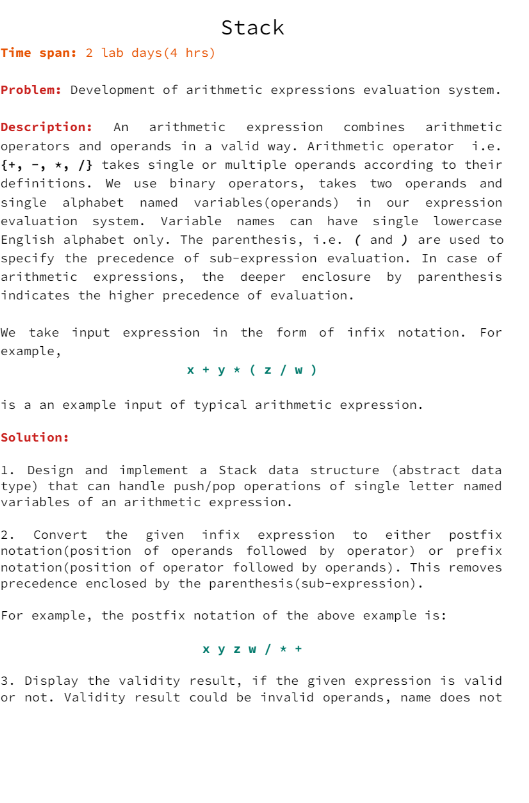
1. **Problems**



**2. Source code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define FALSE 0

#define TRUE 1

#define FAMILY 2

#define INDIVIDUAL 3

#define EMPTYQ -1

#define EMPTYC '\0'

const int MAXSIZE =8;

const int MAXTICKET = 4;

//how to calculate time difference ----------------------------------------------------\*\*

struct Ticket{

    int ticketNum;

    int ticketType;

    char ph[15];

    int arrivalTime;    //WHY USE LONG ARRIVALTIME?

};

struct Queue{

    struct Ticket \*tickets;

    int count;

    int head;

    int tail;

};

int getSize(char \*u){

    int i =0;

    while(u[i] != '\0'){i++;}

    return i;

}

int isQueueFull(struct Queue \*q){

    if((q->tail) +1 ==MAXTICKET) return TRUE;

    else return FALSE;

}

int isQueueEmpty(struct Queue \*q){

    if(q->head > q->tail) return TRUE;

    if(q->head == -1 && q->tail == -1) return TRUE;

    return FALSE;

}

int enqueue(struct Queue \*q,char\* c,int ticketNum,int ticketType,int arrivalTime){

    if(isQueueFull(q)) return FALSE;

    if(checkRepeatPh(q,q,c)) return FALSE;

    if(q->tail == -1 && q->head == -1) q->tail = q->head =0;

    else q->tail++;

    int sizec = getSize(c);

    for(int i=0;i<sizec;i++) q->tickets[q->tail].ph[i] = c[i];

    q->tickets[q->tail].ticketNum=ticketNum;

    q->tickets[q->tail].arrivalTime = arrivalTime;

    q->tickets[q->tail].ticketType = ticketType;

    q->count++;

    return TRUE;

}

void dequeue(struct Queue \*q){

    if(isQueueEmpty(q)) return;

    //these commented out can help queue maintain storage

    //int size1 = getSize(q->tickets);

    //for(int i=0;i<size1-1;i++) q->input[i] = q->input[i+1];

    //q->tail--;

    q->head++;

}

void displayQueue(struct Queue \*q){

    for(int i = q->head;i<=q->tail;i++){

        printf("Phone num : %s \n",q->tickets[i].ph);

        printf("Ticket Num: %d \n",q->tickets[i].ticketNum);

        printf("Arrival Time :%d \n",q->tickets[i].arrivalTime);

        if(q->tickets[i].ticketType == FAMILY) printf("Ticket Type : Family\n");

        else printf("Ticket Type : Individual\n");

        printf("\n\n");

    }

    printf("\n");

}

void initializeQueue(struct Queue \*q){

    q->tickets = (struct Ticket\*) malloc (sizeof(struct Ticket)\*MAXTICKET);

    q->head = q->tail = -1;

    q->count =0;

}

int TotAvailableSeats(struct Queue \*q1,struct Queue \*q2){

    return (MAXSIZE - q1->count - q2->count);

}

int checkRepeatPh(struct Queue \*q1,struct Queue \*q2, char \*ph){

    int found =FALSE;

    for(int i =0;i<q1->count;i++){

        if(strcmp(q1->tickets[i].ph,ph)==0){

            found = TRUE;

            return found;

        }

    }

    for(int i =0;i<q2->count;i++){

        if(strcmp(q2->tickets[i].ph,ph)==0){

            found = TRUE;

            return found;

        }

    }

    return found;

}

int genticketNum(struct Queue \*q1, struct Queue \*q2){

    return q1->count + q2->count+1;

}

int giveticketNum(struct Queue \*q1,struct Queue \*q2,char \*ch){

    int num;

    num=0;

    //printf("%d is count check 1\n",q1->count);

    for(int i =0;i<q1->count;i++){

        if(strcmp(q1->tickets[i].ph,ch)==-10){

            num = q1->tickets[i].ticketNum;

            return num;

        }

    }

    //printf("%d is count check 2\n",q2->count);

    for(int i =0;i<q2->count;i++){

        //printf("%d check 3 for [%d]\n",strcmp(q2->tickets[i].ph,ch),i+1);

        if(strcmp(q2->tickets[i].ph,ch)==-10){

            num = q2->tickets[i].ticketNum;

            return num;

        }

    }

    return num;

}

int giveticketType(struct Queue \*q1,struct Queue \*q2, char \*ch){

    int found = 0;

    for(int i =0;i<q1->count;i++){

        if(strcmp(q1->tickets[i].ph,ch)==0){

            found = q1->tickets[i].ticketType;

            return found;

        }

    }

    for(int i =0;i<q2->count;i++){

        if(strcmp(q2->tickets[i].ph,ch)==0){

            found = q2->tickets[i].ticketType;

            return found;

        }

    }

}

int giveTimeDiff(struct Queue \*q1,struct Queue \*q2){

    int a;

    return a;

}

void input(struct Queue \*qfamily,struct Queue \*qindiv){

    int a;

    a = enqueue(qfamily,"9849023236",genticketNum(qfamily,qindiv),INDIVIDUAL,10);

    a = enqueue(qindiv, "9849023260",genticketNum(qfamily,qindiv),FAMILY,21);

    a = enqueue(qindiv, "9849023299",genticketNum(qfamily,qindiv),FAMILY,22);

    a = enqueue(qindiv, "9849023220",genticketNum(qfamily,qindiv),INDIVIDUAL,23);

    a = enqueue(qfamily,"9849023243",genticketNum(qfamily,qindiv),FAMILY,27);

    //a = enqueue(qindiv,"989890",genticketNum(qfamily,qindiv),FAMILY,55);

    printf("[FAMILY TICKETS: ]\n");

    displayQueue(qfamily);

    printf("[INDIVIDUAL TICKETS:] \n");

    displayQueue(qindiv);

    char temps[15];

    printf("Total Available Seats = %d\n",TotAvailableSeats(qfamily,qindiv));

    printf("Enter which phone number you want to retrieve information of :\n");

    fgets(temps,15,stdin);

    if(giveticketNum(qfamily,qindiv,temps)==0) printf("NOT RESERVED \n");

    else printf("Ticket Number : %d \n",giveticketNum(qfamily,qindiv,temps));

    if((giveticketType(qfamily,qindiv,temps))==3) printf("Ticket type : Family\n");;

}

void cleanMem(struct Queue \*q){

    while(!isQueueEmpty(q)) dequeue(q);

    //free(q->input);

    free(q);

}

int main(){

    struct Queue \*qfamily = (struct Queue\*) malloc(sizeof(struct Queue)\*1);

    struct Queue \*qindiv = (struct Queue\*) malloc(sizeof(struct Queue)\*1);

    initializeQueue(qfamily);

    initializeQueue(qindiv);

    input(qfamily,qindiv);

    cleanMem(qfamily);

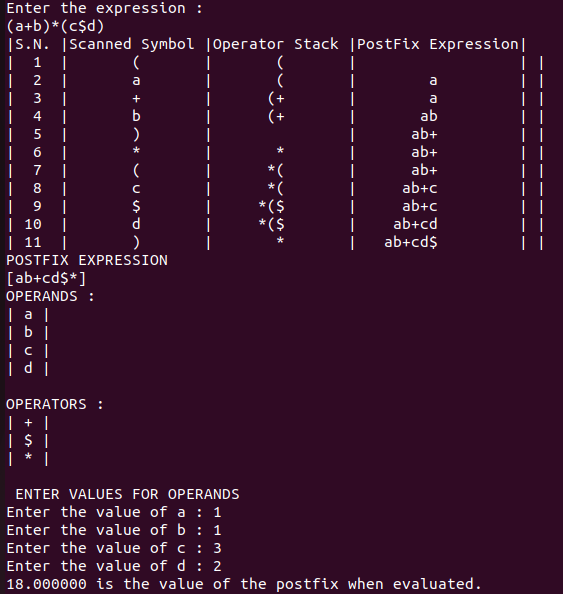
    cleanMem(qindiv);

    return 0;

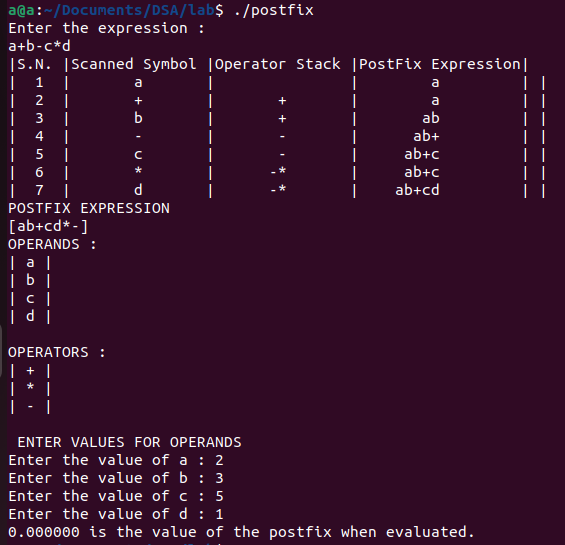
}

Output:-

When (a+b)\*(c$d) is given input to program then the result will be the following:



When a+b-c\*d is given input the following is the result of the program:



**3. Conclusion**

The problems given in LabSheet-2 has made me understand how to develop arithmetic expressions evaluation system and how to implement them while programming in C++/ C. It has made me able to understand the conversion of infix to postfix and it’s evaluation with the help of arithmetic expressions in C++ / C. I was able to understand Stack in a deeper view. Hence I have become familiar with creation of stack which helped in conversion of infix to postfix in C++/ C.