College of Engineering, Thiruvananthapuram

Data Structures Lab

Anirudh A. V.
S2 CSE R2, Roll No. 11

Department of Computer Science
& Engineering

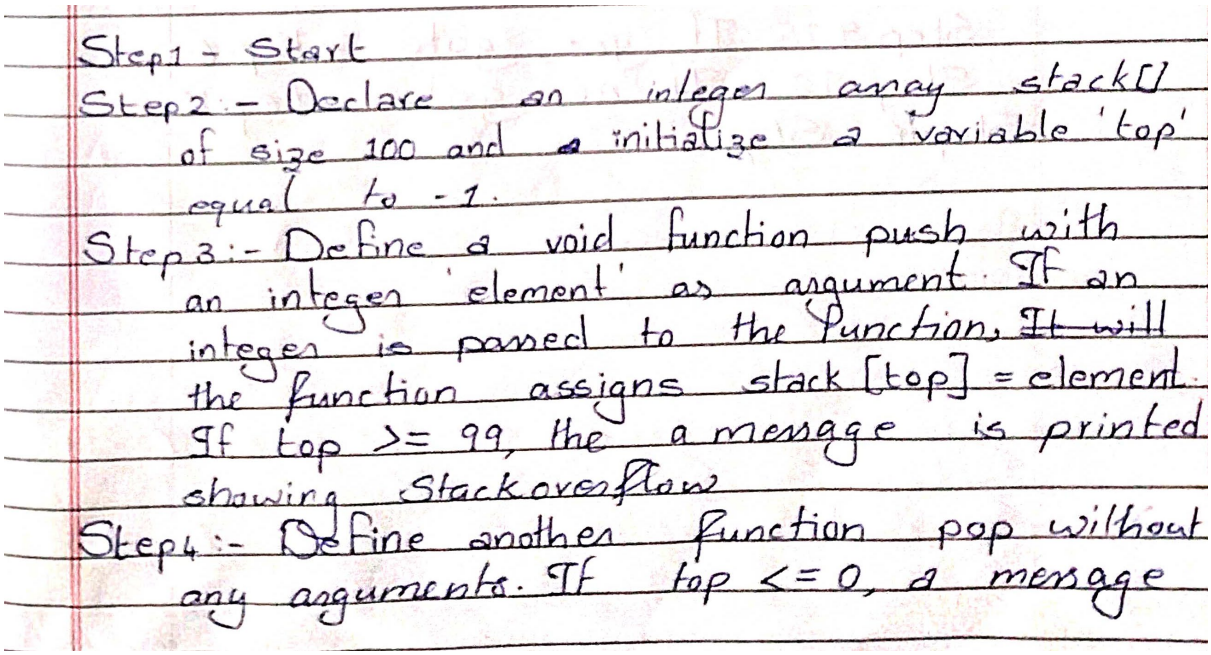
January 19, 2022

1 Stack

1.1 Aim

Write a menu-driven C Program to Implement a Stack using arrays with the
operations:
a. Pushing elements to the Stack.
b. Popping elements from the Stack
c. Display the contents of the Stack after each operation

1.2 Algorithm


Step1 - Start
Step2 - Declare an integer array stack[] of size 100 and initialize a variable 'top' equal to -1.
Step3 :- Define a void function push with an integer 'element' as argument. If an integer is passed to the function, It will the function assigns stack[top] = element. If top >= 99, the a message is printed showing Stack overflow
Step4 :- Define another function pop without any arguments. If top <= 0, a message

showing Stackinflow is printed. else top is decremented.

Step5:- Define a function display without any arguments. Using a for loop, iterating from top to 0 print all the elements of the stack.

Step6:- ~~Display~~ In the main function, Display a menu showing the operations to be performed

Step7:- If choice = 1, ~~Top~~ use push function with the argument as the user input. ~~goto step~~ Display @ stack[] using ~~Step~~ display function. Goto step 9

Step8:- If choice = 2, use pop function to delete the top element and display stack[]. Goto step 9.

Step9:- Display a question whether to continue or not.

Step 9.1:- If yes, goto step6

Step 9.2:- If no, goto step10

Step10:- Stop

1.3 Code

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

#define SIZE 100
int stack[SIZE], n, b, top = -1;

void push(int element);
void pop();
void display();

void main()
{

    printf("Enter the size of the stack : ");
    scanf("%d", &n);

    int response, choice, element;
    do
    {
        printf("\n\nM E N U\n\n1. Pushing elements to the Stack\n2. Popping
elements from the Stack\n3. Exit\n\n");
    ch:
        printf("\t  -> ");
        scanf("%d", &choice);
        printf("\n\n");
        switch (choice)
        {
        case 1:
            printf("Enter the element to be pushed : ");
            scanf("%d", &element);
            push(element);
            display();
            break;

        case 2:
            pop();
            display();
            break;

        case 3:
            exit(0);

        default:
            printf("Enter a valid Choice!!!!\n\n");
            goto ch;
        }
        printf("\nDo you want to continue (1 or 0) : ");
```

```
        scanf("%d", &response);
    } while (response == 1);
}

void push(int element)
{
    if (top >= n - 1)
    {
        printf("Stack Overflow!!!\n\n");
        return;
    }
    else
    {
        top++;
        stack[top] = element;
        printf("\n\n");
    }
}

void pop()
{
    if (top <= 0)
    {
        printf("\nStack Inflow!!!\n");
    }
    else
    {
        top--;
    }
}

void display()
{
    printf("Stack : \n");
    for (int i = top; i >= 0; i--)
    {
        printf("\t  %d\n", stack[i]);
    }
}
```

1.4 Sample Output

```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cy
cle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.
exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\U
sers\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4
c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin
 Prime "

17 is a prime number.

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l>
```

2 Infix to Postfix Conversion

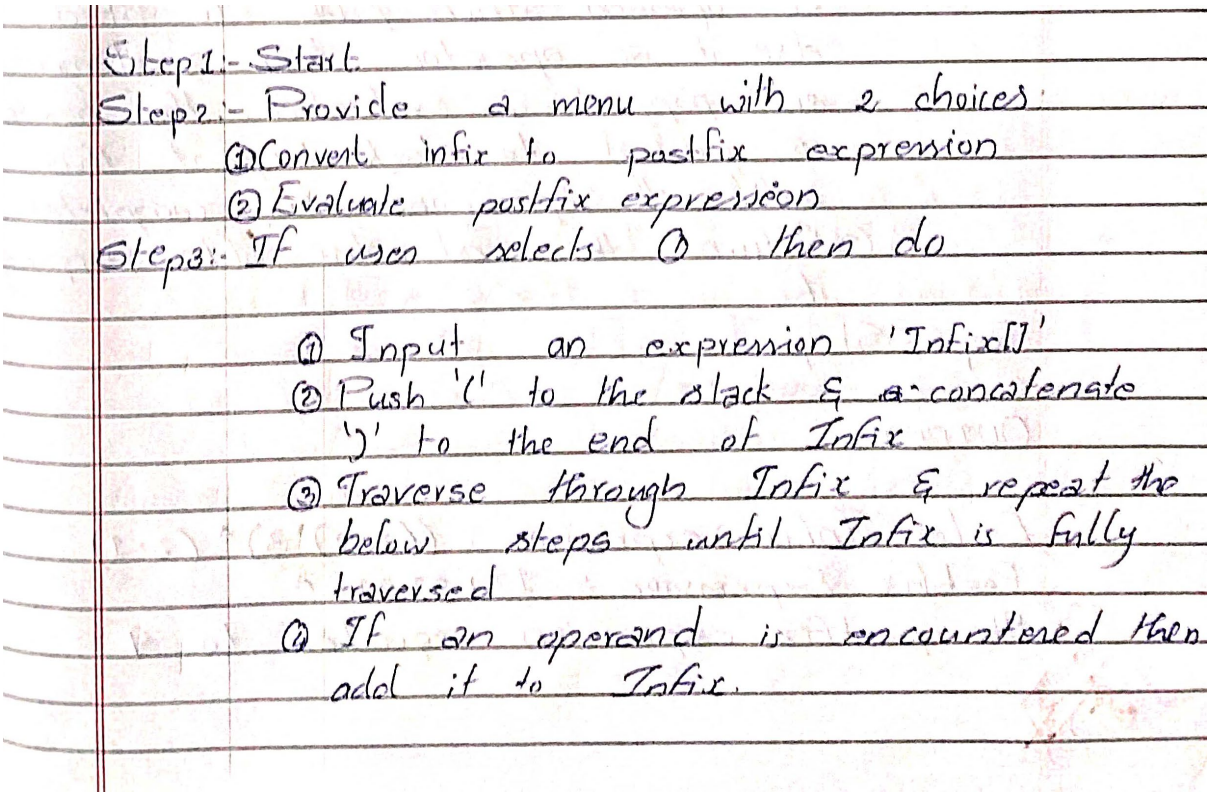
2.1 Aim

Write a menu driven C Program, to do the following operations using a stack data structure:

a) Convert an infix expression to a postfix expression

b) Evaluate the postfix expression

2.2 Algorithm



⑤ If left parenthesis '(' then push to stack

⑥ If an operator is encountered, Check priority order of it with the operator at top of stack. If greater or equal pop until less priority is seen or '('. Else push operator to stack

⑦ If ')' is encountered, then repeatedly pop from the stack until left parenthesis is encountered, remove left parenthesis

⑧ Return postfix expression

Step4:- If option ② is selected, then:

① Input the values to be evaluated

② Traverse through the expression, if an operand then push to stack, else if an operator then 2 elements are popped & evaluated, the result is pushed to stack. Repeat this until all operations are completed.

③ Return the end value after popping it.

Step5:- Stop

2.3 Code

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

#define SIZE 100

char stack[SIZE];
int top = -1;

void push(char item)
{
    if (top >= SIZE - 1)
    {
        printf("\nStack Overflow.\n");
    }
    else
    {
        top = top + 1;
        stack[top] = item;
    }
}

char pop()
{
    char item;

    if (top < 0)
    {
        printf("stack under flow: invalid infix expression");
        getchar();
        exit(1);
    }
    else
    {
        item = stack[top];
        top = top - 1;
        return (item);
    }
}

int is_operator(char symbol)
{
    if (symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' ||
symbol == '-')
    {
        return 1;
```

```
    }
    else
    {
        return 0;
    }
}

int precedence(char symbol)
{
    if (symbol == '^')
    {
        return (3);
    }
    else if (symbol == '*' || symbol == '/')
    {
        return (2);
    }
    else if (symbol == '+' || symbol == '-')
    {
        return (1);
    }
    else
    {
        return (0);
    }
}

void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
    int i, j;
    char item, x;

    push('(');
    strcat(infix_exp, ")");

    i = 0;
    j = 0;
    item = infix_exp[i];

    while (item != '\0')
    {
        if (item == '(')
        {
            push(item);
        }
        else if (isdigit(item) || isalpha(item))
        {
            postfix_exp[j] = item;
```

```
            j++;
        }
        else if (is_operator(item) == 1)
        {
            x = pop();
            while (is_operator(x) == 1 && precedence(x) >= precedence(item))
            {
                postfix_exp[j] = x;
                j++;
                x = pop();
            }
            push(x);
            push(item);
        }
        else if (item == ')')
        {
            x = pop();
            while (x != '(')
            {
                postfix_exp[j] = x;
                j++;
                x = pop();
            }
        }
        else
        {
            printf("\nInvalid infix Expression.\n");
            getchar();
            exit(1);
        }
        i++;
        item = infix_exp[i];
    }
    if (top > 0)
    {
        printf("\nInvalid infix Expression.\n");
        putchar(item);
        exit(1);
    }
    postfix_exp[j] = '\0';
}

int calculation(int a, int b, char O)
{
    int mult = 1;
    if (O == '+')
    {
        return a + b;
```

```
    }
    else if (O == '-')
    {
        return a - b;
    }
    else if (O == '*')
    {
        return a * b;
    }
    else if (O == '/' && b != 0)
    {
        return a / b;
    }
    else if (O == '^')
    {
        for (int i = 1; i <= b; i++)
        {
            mult = a * mult;
        }
        return mult;
    }
}
int evaluation(char postfix_exp[])
{
    char a;
    int i = 0, result;
    do
    {
        if (isdigit(postfix_exp[i]))
        {
            push(postfix_exp[i]);
        }
        else if (is_operator(postfix_exp[i]))
        {
            a = pop();
            push(calculation(pop() - '0', a - '0', postfix_exp[i]) + '0');
        }
        else if (isalpha(postfix_exp[i]))
        {
            printf("The expression contains a variable!!!");
            getchar();
            exit(1);
        }
        else
        {
            printf("\nInvalid Expression!!!\n\n");
            getchar();
            exit(1);
```

```
        }
        i++;
    } while (postfix_exp[i] != '\0');
    return pop();
}

int main()
{
    char infix[SIZE], postfix[SIZE];

    printf("\nEnter Infix expression : ");
    gets(infix);
    int response, choice, element;
    do
    {
        printf("\n\nM E N U\n\n1. Convert the Infix Expression into
Postfix\n2. Evaluate the postfix expression\n3. Exit\n\n");
    ch:
        printf("\t  -> ");
        scanf("%d", &choice);
        printf("\n\n");
        switch (choice)
        {
        case 1:
            InfixToPostfix(infix, postfix);
            printf("\nPostfix Expression: ");
            puts(postfix);
            break;

        case 2:
            printf("\nThe postfix expression equates to %c\n",
evaluation(postfix));
            break;

        case 3:
            exit(0);

        default:
            printf("Enter a valid Choice!!!!\n\n");
            goto ch;
        }
        printf("\nDo you want to continue (1 or 0) : ");
        scanf("%d", &response);
    } while (response == 1);

    return 0;
}
```

2.4 Sample Output

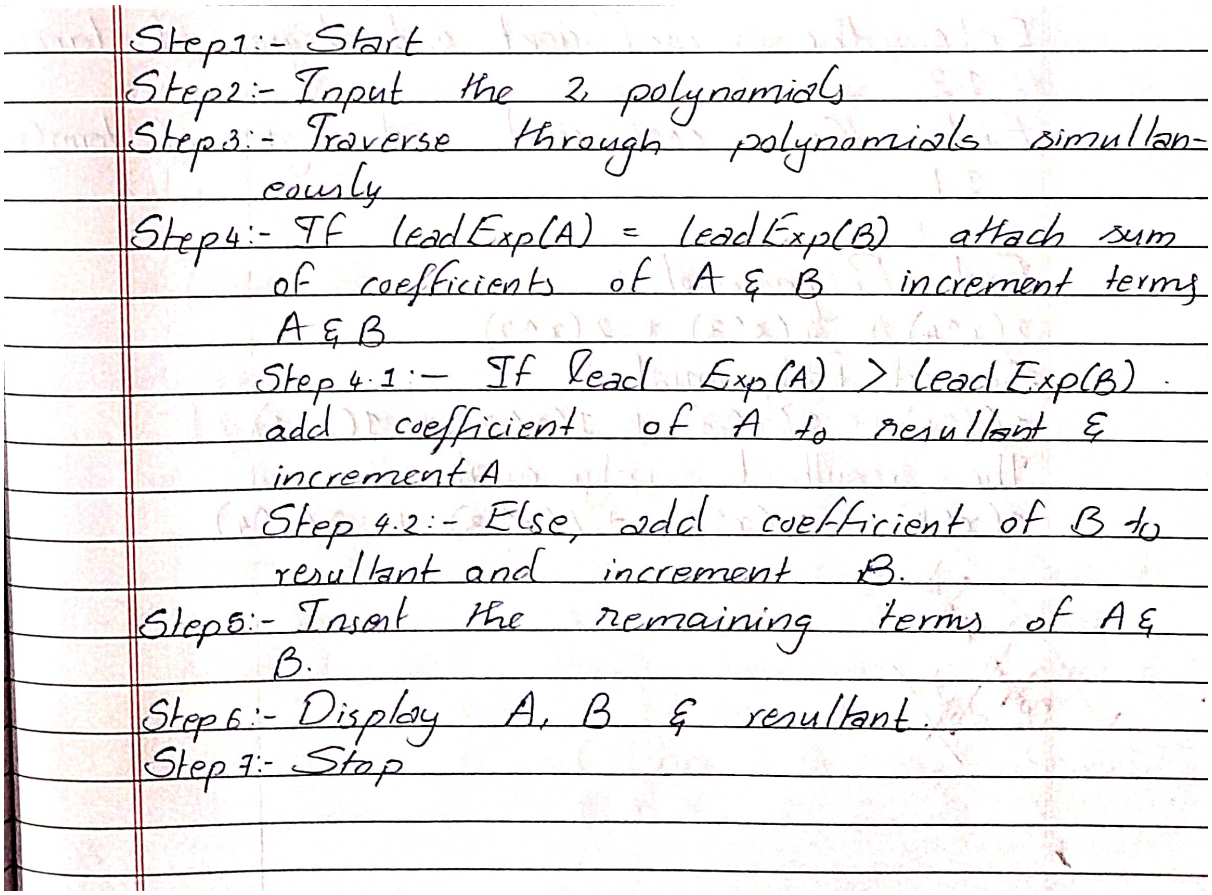
```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cy
cle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.
exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\U
sers\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4
c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin
 palindrome "

malayalam is a palindrome.

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l>
```

3 Polynomial

3.1 Aim

Write a program to read two polynomials and store them in an array. Calculate the sum of the two polynomials and display the first polynomial, second polynomial, and the resultant polynomial.

3.2 Algorithm



3.3 Code

```
#include <stdio.h>
#include <math.h>

#define MAX_TERMS 30

typedef struct polynomial
{
    int coef;
    int expon;
} polynomial;

polynomial A[MAX_TERMS], B[MAX_TERMS], C[MAX_TERMS];

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////

int Add(struct polynomial A[30], struct polynomial B[30], int termsA, int
termsB, struct polynomial C[30]);
void displayPoly(struct polynomial p[30], int term);

void swap(int *p, int *q)
{
    int temp;
    temp = *p;
    *p = *q;
    *q = temp;
}

void sort(struct polynomial C[30], int termsC)
{
    for (int i = 0; i < termsC; i++)
    {
        for (int j = i + 1; j < termsC; j++)
        {
            if (C[i].expon < C[j].expon)
            {
                swap(&C[i].expon, &C[j].expon);
                swap(&C[i].coef, &C[j].coef);
            }
        }
    }
}

////////////////////////////////////////////////////////////////////////////
/////////////////////////////////////////

void main()
```

```
{
    int termsA, termsB, termsC;

    printf("\nEnter the number of terms of the first polynomial : ");
    scanf("%d", &termsA);

    for (int i = 0; i < termsA; i++)
    {
        printf("Enter the coefficient and degree of term (%d) : ", i + 1);
        scanf("%d%d", &A[i].coef, &A[i].expon);
    }

    printf("Enter the number of terms of the second polynomial : ");
    scanf("%d", &termsB);

    for (int i = 0; i < termsB; i++)
    {
        printf("Enter the coefficient and degree of term (%d) : ", i + 1);
        scanf("%d%d", &B[i].coef, &B[i].expon);
    }

    printf("\nFirst Polynomial : \n");
    displayPoly(A, termsA);

    printf("\nSecond Polynomial : \n");
    displayPoly(B, termsB);

    termsC = Add(A, B, termsA, termsB, C);

    sort(C, termsC);

    printf("\nThe resultant polynomial : \n");
    displayPoly(C, termsC);
}

//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////

int Add(struct polynomial A[30], struct polynomial B[30], int termsA, int
termsB, struct polynomial C[30])
{
    int i = 0, j = 0, k = 0;

    while (i < termsA || j < termsB)
    {
        if (A[i].expon == B[j].expon)
        {
            C[k].coef = A[i].coef + B[j].coef;
```

```
            C[k].expon = A[i].expon;
            i++;
            j++;
            k++;
        }
        else if (A[i].expon > B[j].expon)
        {
            C[k].coef = A[i].coef;
            C[k].expon = A[i].expon;
            i++;
            k++;
        }
        else if (A[i].expon < B[j].expon)
        {
            C[k].coef = B[j].coef;
            C[k].expon = B[j].expon;
            j++;
            k++;
        }
    }
    if (j >= termsB)
    {
        while (i < termsA)
        {
            C[k].coef = A[i].coef;
            C[k].expon = A[i].expon;
            i++;
            k++;
        }
    }
    if (i >= termsA)
    {
        while (j < termsB)
        {
            C[k].coef = B[j].coef;
            C[k].expon = B[j].expon;
            j++;
            k++;
        }
    }
    return (k);
}

void displayPoly(struct polynomial p[30], int term)
{
    for (int i = 0; i < term - 1; i++)
    {
        printf("%d(x^%d) + ", p[i].coef, p[i].expon);
```

```
    }
    printf("%d(x^%d)", p[term - 1].coef, p[term - 1].expon);
}
```

3.4 Sample Output

```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_official>
e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_
official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.exe" --en
able-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\vinod\A
ppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4c5cd1cd090876b1
a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin Frequency "

'a' occurs 5 times in 'Hello, I am Baymax. It is a pleasure to meet you'.


E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_official>
```

4 Sparse Matrix

4.1 Aim

Write a C program to enter two matrices in normal form Do the following operations, implemented as separate functions :

a) Convert two matrices to tuple form and display it.
b) Find the transposes of the two matrices represented in tuple form and display them in tuple form.
c) Find the sum of the two matrices in tuple form and display the sum in tuple form.

4.2 Algorithm

i) Start
ii) Read the elements of the matrix
iii) Traverse through the elements of 2D array
iv) If the element is non-zero store its row no, col. no and value in an array structures, with fields row num, col. num & value
v) Increment 1 and move to next element of matrix.

vi) Store the total no. of rows, & columns & the total no. of non-zero value in the first element of array of structures.

vii) To find the transpose of the matrix do the following:

a) Start

b) A be the original tuple form of matrix & T be the transpose. Let row_A & col_A denote no. of rows & columns of A & $rows_T$ & col_T be row no.s & col nos of transpose, T. Let n be the no. of non zero elements.

c) Let $row_T = col_A$ & $col_T = row_A$, $value_T = n$.

d) Let t be the current index of T. Initialise t with 1.

e) For i from $0 \longrightarrow col_A$ do (f)

f) For j from $1 \longrightarrow n$ do (g)

g) If A[j].col = i, then assign

```
T[t].row = A[j].col
T[t].col = A[j].row
T[t].value = A[j].value
Increment t by 1.
```

h) Stop

viii) To find the addition result of 2 matrices:

FABER-CASTELL
Date
Page No.

a) Start
b) Let input matrices in tuple form be A & B. The resultant matrix in row, column format be C
c) Check whether A[0].row = B[0].row & A[0].column = B[0].column, else stop.
d) Move to next term in both A & B
e) Let $< row_a, col_a, val_a >$ denote content term A and $< row_b, col_b, val_b >$, denote in B, while there is unprocessed non-zero terms in Both A & B. repeat from f to h.
f) If ($row_a = row_b$ & $col_a = col_b$) then assign $C[row_a][col_a] = val_a + val_b$ move to next term in both A & B.
g) If ($row_b < row_a$) or ($row_b = row_a$ and $col_a < col_b$) then assign $C[row_a][col_b] = val_a$, move to next term in A.
h) If ($row_b < row_a$) or [($row_b = row_a$) & ($col_b < col_a$)] then assign $C[row_b][col_b] = val_b$, move to next term in B.
i) If there is unprocessed non-zero terms in A/B assign the unprocessed terms to the corresponding location of resultant matrix.
j) Stop

4.3 Code

```
#include <stdio.h>

#define MAX_TERMS 30

typedef struct tuple
{
    int row;
    int column;
    int value;
} tuple;

tuple A[MAX_TERMS], B[MAX_TERMS], C[MAX_TERMS];

void sort_tuple(struct tuple tuple_form[], int n, int no_of_elements)
{
    struct tuple temp;
    for (int i = 0; i < no_of_elements; i++)
    {
        for (int j = i + 1; j < no_of_elements; j++)
        {
            if (n == 1)
            {
                if (tuple_form[i].row > tuple_form[j].row)
                {
                    temp = tuple_form[i];
                    tuple_form[i] = tuple_form[j];
                    tuple_form[j] = temp;
                }
                else if ((tuple_form[i].row == tuple_form[j].row) &&
(tuple_form[i].column > tuple_form[j].column))
                {
                    temp = tuple_form[i];
                    tuple_form[i] = tuple_form[j];
                    tuple_form[j] = temp;
                }
            }
            if (n == 2)
            {
                if (tuple_form[i].column > tuple_form[j].column)
                {
                    temp = tuple_form[i];
                    tuple_form[i] = tuple_form[j];
                    tuple_form[j] = temp;
                }
                else if ((tuple_form[i].column == tuple_form[j].column) &&
(tuple_form[i].row > tuple_form[j].row))
                {
```

```
                    temp = tuple_form[i];
                    tuple_form[i] = tuple_form[j];
                    tuple_form[j] = temp;
                }
            }
        }
    }
}

void Display_matrix(int matrix[][20], int row, int col)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

void Read_matrix(int matrix[][20], int row, int col)
{
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            scanf("%d", &matrix[i][j]);
        }
    }
}

int Conversion(int matrix[][20], int row, int col, struct tuple tuple_form[])
{
    int k = 0;
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            if (matrix[i][j] != 0)
            {
                tuple_form[k].row = i;
                tuple_form[k].column = j;
                tuple_form[k].value = matrix[i][j];
                k++;
            }
        }
    }
```

```
    return k;
}

void display_tuple(struct tuple tuple_form[], int no_of_elements, int row, int
column)
{
    sort_tuple(tuple_form, 1, no_of_elements);
    printf("(\trow \tcol \tval )\n");
    printf("( \t %d \t %d \t %d )\n\n", row, column, no_of_elements);
    for (int i = 0; i < no_of_elements; i++)
    {
        if (tuple_form[i].value != 0)
        {
            printf("( \t %d \t %d \t %d )\n", tuple_form[i].row,
tuple_form[i].column, tuple_form[i].value);
        }
    }
}

void Transpose(struct tuple tuple_form[], int no_of_elements, int row, int
column)
{
    sort_tuple(tuple_form, 2, no_of_elements);
    printf("(\trow \tcol \tval )\n");
    printf("( \t %d \t %d \t %d )\n\n", column, row, no_of_elements);
    for (int i = 0; i < no_of_elements; i++)
    {
        printf("( \t %d \t %d \t %d )\n", tuple_form[i].column,
tuple_form[i].row, tuple_form[i].value);
    }
    printf("\n");
}

int Add(struct tuple A[], int TupleA, struct tuple B[], int TupleB, struct
tuple C[])
{
    int i = 0, j = 0, k = 0;
    sort_tuple(A, 1, TupleA);
    sort_tuple(B, 1, TupleB);

    while (i < TupleA && j < TupleB)
    {
        if (A[i].row == B[j].row && A[i].column == B[j].column)
        {
            C[k].value = A[i].value + B[j].value;
            C[k].row = A[i].row;
            C[k].column = A[i].column;
            i++;
```

```
            j++;
            k++;
        }
        else if ((A[i].row < B[j].row) || (A[i].row == B[j].row && A[i].column
< B[j].column))
        {
            C[k].value = A[i].value;
            C[k].row = A[i].row;
            C[k].column = A[i].column;
            i++;
            k++;
        }
        else if ((A[i].row > B[j].row) || (A[i].row == B[j].row && A[i].column
> B[j].column))
        {
            C[k].value = B[j].value;
            C[k].row = B[j].row;
            C[k].column = B[j].column;
            j++;
            k++;
        }
    }
    if (j >= TupleB)
    {
        while (i < TupleA)
        {
            C[k].value = A[i].value;
            C[k].row = A[i].row;
            C[k].column = A[i].column;
            i++;
            k++;
        }
    }
    if (i >= TupleA)
    {
        while (j < TupleB)
        {
            C[k].value = B[j].value;
            C[k].row = B[j].row;
            C[k].column = B[j].column;
            j++;
            k++;
        }
    }
    return (k);
}

void main()
```

```
{
    int m1, m2, n1, n2, tupleA, tupleB, tupleC;

    printf("Enter the order of first matrix (mxn) : ");
    scanf("%d%d", &m1, &n1);
    printf("Enter the order of second matrix (mxn) : ");
    scanf("%d%d", &m2, &n2);

    int matrixA[10][20], matrixB[10][20];

    printf("\nEnter the first matrix : ");
    Read_matrix(matrixA, m1, n1);

    printf("\nEnter the second matrix : ");
    Read_matrix(matrixB, m2, n2);

    printf("\nMatrix A : \n");
    Display_matrix(matrixA, m1, n1);

    printf("\nMatrix B : \n");
    Display_matrix(matrixB, m2, n2);

    int response, choice;
    do
    {
        printf("\n\n\tM E N U\n\n");
        printf("1. Convert the normal matrix into tuple form\n2. Find the
transpose of two matrices\n3. Add the two matrices\n\n");
    ch:
        printf("\t->");
        scanf("%d", &choice);
        printf("\n");

        switch (choice)
        {
        case 1:
            tupleA = Conversion(matrixA, m1, n1, A);
            tupleB = Conversion(matrixB, m2, n2, B);
            printf("\nMatrix A : \n");
            display_tuple(A, tupleA, m1, n1);
            printf("\nMatrix B : \n");
            display_tuple(B, tupleB, m2, n2);
            break;

        case 2:
            printf("\nTranspose of Matrix A : \n");
            Transpose(A, tupleA, m1, n1);
            printf("\nTransponse of Matrix B : \n");
```

```
            Transpose(B, tupleB, m2, n2);
            break;

        case 3:
            if (m1 == m2 && n1 == n2)
            {
                tupleC = Add(A, tupleA, B, tupleB, C);
                display_tuple(C, tupleC, m1, n1);
            }
            else
            {
                printf("\nIncompatible Matrices!!!\n\n");
            }
            break;

        default:
            printf("\nEnter a valid option!!!\n");
            goto ch;
        }

        printf("\nDo you want to continue(1 or 0) : ");
        scanf("%d", &response);

    } while (response == 1);
}
```

4.4 Sample Output

```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_offic
ial> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Jav
a_cycle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin
\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages
-cp C:\Users\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b7
1a436977c4c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_
704f625c\bin Reverse "

The reversed string : napanamdapahtnanA
```

5 Queue

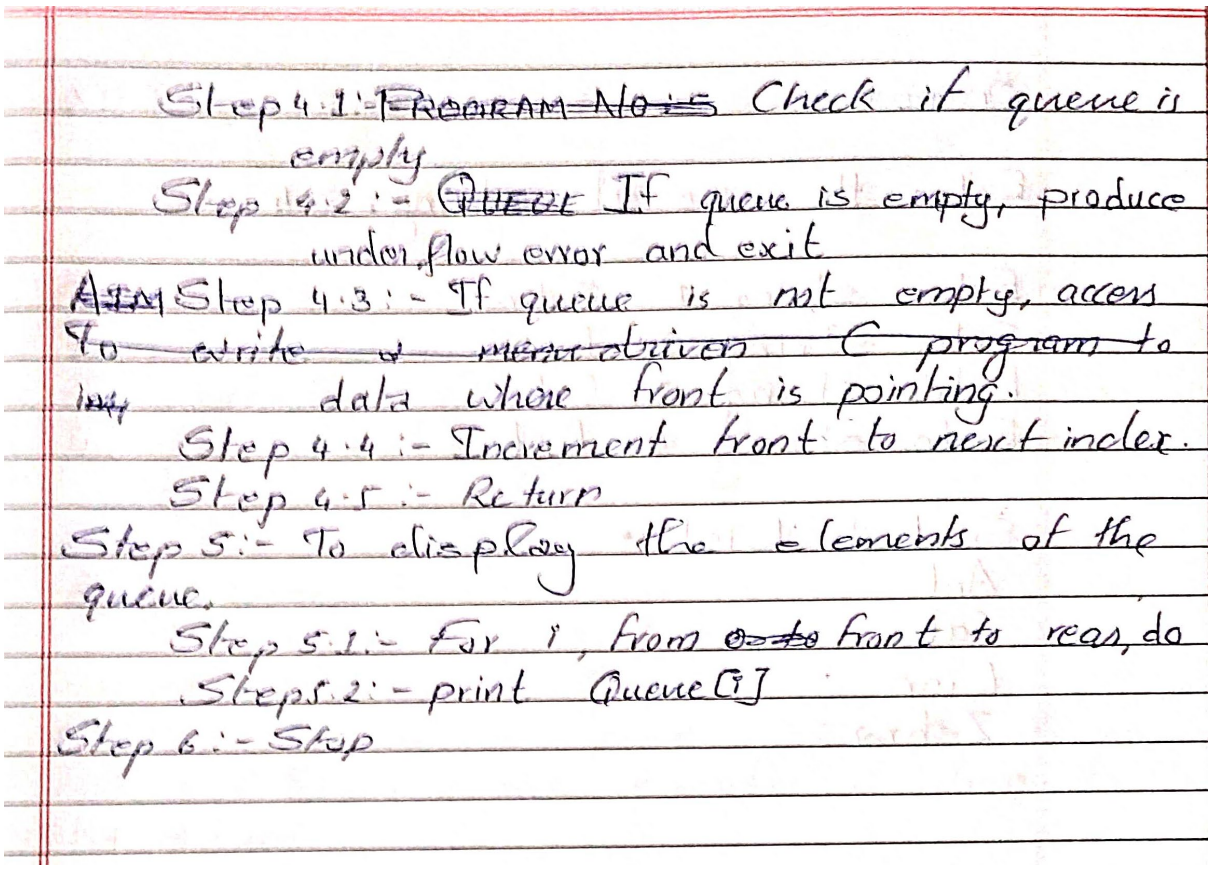
5.1 Aim

Write a menu-driven C Program to implement a Queue using arrays with the following operations:

a. Insert elements to the Queue.
b. Delete elements from the Queue.
c. Display the contents of the Queue after each operation.

5.2 Algorithm

Step 1:- Start
Step 2:- To add elements go to step 3, to delete elements go to step 4, to display go to step 5
Step 3:- Adding elements to the stack
Step 3.1:- Check if queue is full
Step 3.2:- If queue is full, produce overflow error and exit
Step 3.3:- If queue is not full, increment rear to the next empty space
Step 3.4:- Add data element to the queue location where index is rear
Step 3.5:- Return.
Step 4:- To delete elements from the queue



5.3 Code

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

#define MAX_TERMS 101

int Queue[MAX_TERMS];
int front = -1, rear = -1;

void Enqueue(int element)
{
    if (rear == MAX_TERMS - 1)
    {
        printf("\nThe Queue is Full!!!\n");
    }
    else
    {
        if (rear == -1 && front == -1)
        {
            front = 0;
        }
        rear++;
        Queue[rear] = element;
```

```
    }
}

int Delete()
{
    int element;
    if (front < 0)
    {
        printf("\nThe Queue is empty!!!\n");
    }
    else
    {
        element = Queue[front];
        if (front == rear)
        {
            front = 0;
            rear = 0;
        }
        else
        {
            front++;
        }
    }
    return element;
}

void Display()
{
    printf("\nQueue : \n\tFront -> ");
    for (int i = front; i <= rear; i++)
    {
        printf("%d ", Queue[i]);
    }
    printf("<- Rear\n");
}

void main()
{
    int response, choice, element;
    do
    {
        printf("\n\nM E N U\n\n1. Enqueue elements to the Queue.\n2. Delete
elements from the Queue.\n3. Exit\n\n");
    ch:
        printf("\t  -> ");
        scanf("%d", &choice);
        printf("\n\n");
        switch (choice)
```

```
        {
        case 1:
            printf("Enter the element to be Enqueueed : ");
            scanf("%d", &element);
            Enqueue(element);
            Display();
            break;

        case 2:
            Delete();
            Display();
            break;

        case 3:
            exit(0);

        default:
            printf("Enter a valid Choice!!!!\n\n");
            goto ch;
        }
        printf("\nDo you want to continue (1 or 0) : ");
        scanf("%d", &response);
    } while (response == 1);
}
```

5.4 Sample Output

```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_offic
ial> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Jav
a_cycle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin
\java.exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages
-cp C:\Users\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b7
1a436977c4c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_
704f625c\bin Second "

The second smallest element is 12
```

6 Circular Queue

6.1 Aim

Write a menu-driven C Program to implement a circular queue using arrays with the following operations:

a. Insert an element to the queue.
b. Delete an element from the queue.
c. Display the contents of the queue after each operation.

6.2 Algorithm

Step 1 :- Start
Step 2 :- Accept choice from the user for adding an element, go to step 3, for deleting an element goto Step 5, for exiting go to step 7.
Step 3 :- Accept the element to be added.
Step 4 :- the queue and add the elements if it is empty. Otherwise increment rear by (n+1)%n and add element to the rear and got step 6
Step 6 :- If front and rear are not equal, increment front by (n+1)%n. If they are equal, Set front and rear as -1, go to step 5
Step 6 :- For displaying elements, Loop through indices from front to rear and print the element in that order. Goto step 1.
Step 6 :- Stop

6.3 Code

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

#define MAX_TERMS 5

int Queue[MAX_TERMS], front = -1, rear = -1;

int isEmpty()
{
    if (front == -1)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

void Enqueue(int element)
{
    if ((rear == front - 1) || (front == 0 && rear == MAX_TERMS - 1))
    {
        printf("\nThe Queue is Full!!!\n\n");
        return;
    }
    else
    {
        if (rear == -1 && front == -1)
        {
            front = 0;
        }
        rear = (rear + 1) % MAX_TERMS;
        Queue[rear] = element;
    }
}

int Dequeue()
{
    if (front == -1 && rear == -1)
    {
        printf("\nThe Queue is empty!!!\n\n");
    }
    else
    {
        int value = Queue[front];
        if (front == rear)
```

```
        {
            front = -1;
            rear = -1;
        }
        else
        {
            front = (front + 1) % MAX_TERMS;
        }
        return value;
    }
}

void Display()
{
    if (isEmpty())
    {
        printf("\nThe Queue is now empty.\n\n");
    }
    else
    {
        if (rear >= front)
        {
            printf("\nQueue : \n\tFront -> ");
            for (int i = front; i <= rear; i++)
            {
                printf("%d ", Queue[i]);
            }
            printf("<- Rear\n");
        }
        else
        {
            printf("\nQueue : \n\tFront -> ");
            for (int i = front; i < MAX_TERMS; i++)
            {
                printf("%d ", Queue[i]);
            }
            for (int i = 0; i <= rear; i++)
            {
                printf("%d ", Queue[i]);
            }
            printf("<- Rear\n");
        }
    }
}

void main()
{
    int response, choice, element;
```

```
    do
    {
        printf("\n\nM E N U\n\n1. Enqueue elements to the Queue.\n2. Dequeue
elements from the Queue.\n3. Exit\n\n");
    ch:
        printf("\t  -> ");
        scanf("%d", &choice);
        printf("\n\n");
        switch (choice)
        {
        case 1:
            printf("Enter the element to be Enqueueed : ");
            scanf("%d", &element);
            Enqueue(element);
            Display();
            break;

        case 2:
            Dequeue();
            Display();
            break;

        case 3:
            exit(0);

        default:
            printf("Enter a valid Choice!!!!\n\n");
            goto ch;
        }
        printf("\nDo you want to continue (1 or 0) : ");
        scanf("%d", &response);
    } while (response == 1);
}
```

6.4 Sample Output

```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_offic
ial> cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.exe" --enabl
e-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\Users\vinod
\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4c5cd1cd0
90876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin Matri
x "
Matrix A :
1 2 3
4 5 6
7 8 9
Matrix B :
1 0 0
0 1 0
0 0 1
Product :
1 2 3
4 5 6
7 8 9
```

7 DEQUEUE

7.1 Aim

Write a menu-driven C Program to implement a Double-Ended Queue with the following operations:

a. Insert elements to the front of the queue.

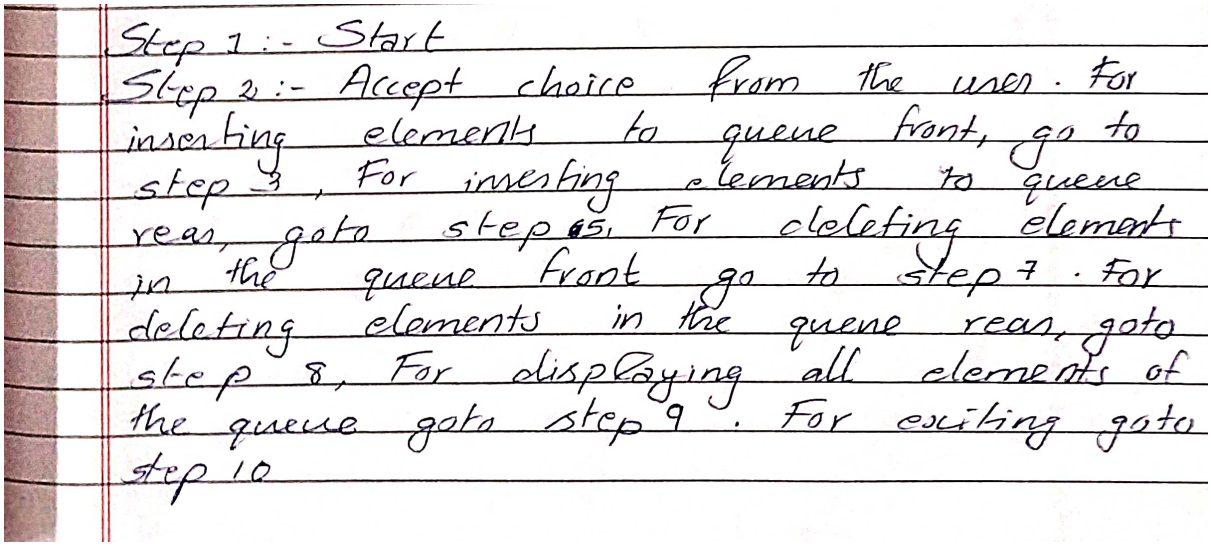
b. Insert elements to the Rear of the queue

c. Delete elements from the front of the queue.

d. Delete elements from the Rear of the queue.

e. Display the queue after each operation.

7.2 Algorithm



Step 3 :- Accept the element from the user.
Step 4 :- Initialize the queue if empty, If queue is not empty, element front by 1, and add element at the front. goto step 9
Step 5 :- Accept the element from the user.
Step 6 :- Initialize the queue if empty. If the queue is not full, increment rear by 1 and add element to the queue from rear goto step 9

Step 7 :- If the queue is not empty, increment front by 1. If front == rear, set front, ~~fro~~ rear as -1, goto step 9
Step 8 :- If queue is not empty, check, if front = rear, If equal, assign front as -1 Else decrement rear by 1. Go to step 9
Step 9 :- Loop from front to rear and display all the elements in the queue. Goto step 2:
Step 10 :- Stop

7.3 Code

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

#define MAX_TERMS 101

int Queue[MAX_TERMS];
int front = -1, rear = -1, terms = 0;

void Insert_rear(int element)
{
    if (rear == MAX_TERMS - 1)
    {
        printf("\nThe Queue is Full!!!\n");
    }
    else
```

```
    {
        if (rear == -1 && front == -1)
        {
            front = 0;
        }
        rear++;
        Queue[rear] = element;
        terms++;
    }
}

void Insert_front(int element)
{
    if (front == 0 && rear == MAX_TERMS - 1)
    {
        printf("\nInsertion not possible!!!\n");
    }
    else if (front == 0)
    {
        int i = terms - 1;
        while (i >= 0)
        {
            Queue[i + 1] = Queue[i];
            i--;
        }
        Queue[front] = element;
        rear++;
        terms++;
        return;
    }
    else
    {
        if (rear == -1 && front == -1)
        {
            front = 0;
            rear = 0;
            Queue[front] = element;
            terms++;
            return;
        }
        front--;
        Queue[front] = element;
        terms++;
    }
}

int Delete_front()
{
```

```
    int element;
    if (front < 0)
    {
        printf("\nThe Queue is empty!!!\n");
    }
    else
    {
        element = Queue[front];
        if (front == rear)
        {
            front = 0;
            rear = 0;
        }
        else
        {
            front++;
        }
        terms--;
    }
    return element;
}

int Delete_rear()
{
    int element;
    if (front < 0)
    {
        printf("\nThe Queue is empty!!!\n");
    }
    else
    {
        element = Queue[rear];
        if (front == rear)
        {
            front = 0;
            rear = 0;
        }
        else
        {
            if (rear != 0)
            {
                rear--;
            }
        }
        terms--;
    }
    return element;
}
```

```
void Display()
{
    printf("\nQueue : \n\tFront -> ");
    for (int i = front; i <= rear; i++)
    {
        printf("%d ", Queue[i]);
    }
    printf("<- Rear\n");
}

void main()
{
    int response, choice, element;
    do
    {
        printf("\n\nM E N U\n\n1. Insert elements to the Front of the
queue.\n2. Insert elements to the Rear of the queue.\n");
        printf("3. Delete elements from the Front of the queue.\n4. Delete
elements from the Rear of the queue.\n5. Exit\n\n");
    ch:
        printf("\t  -> ");
        scanf("%d", &choice);
        printf("\n\n");
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted : ");
            scanf("%d", &element);
            Insert_front(element);
            Display();
            break;

        case 2:
            printf("Enter the element to be inserted : ");
            scanf("%d", &element);
            Insert_rear(element);
            Display();
            break;

        case 3:
            Delete_front();
            Display();
            break;

        case 4:
            Delete_rear();
            Display();
```

```
            break;

        case 5:
            exit(0);

        default:
            printf("Enter a valid Choice!!!!\n\n");
            goto ch;
        }
        printf("\nDo you want to continue (1 or 0) : ");
        scanf("%d", &response);
    } while (response == 1);
}
```

7.4 Sample Output

```
E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l> e: && cd "e:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cy
cle_1_official" && cmd /C ""c:\Program Files\Java\jdk-17.0.1\bin\java.
exe" --enable-preview -XX:+ShowCodeDetailsInExceptionMessages -cp C:\U
sers\vinod\AppData\Roaming\Code\User\workspaceStorage\9dd9b71a436977c4
c5cd1cd090876b1a\redhat.java\jdt_ws\Java_cycle_1_official_704f625c\bin
 Transpose "

Matrix :
1 2 3
4 5 6
7 8 9

Transpose :
1 4 7
2 5 8
3 6 9

E:\Anirudh\Anirudh\CET\SEM 3\OOP_Lab\Java_cycle_1\Java_cycle_1_officia
l>
```

8 Priority Queue

8.1 Aim

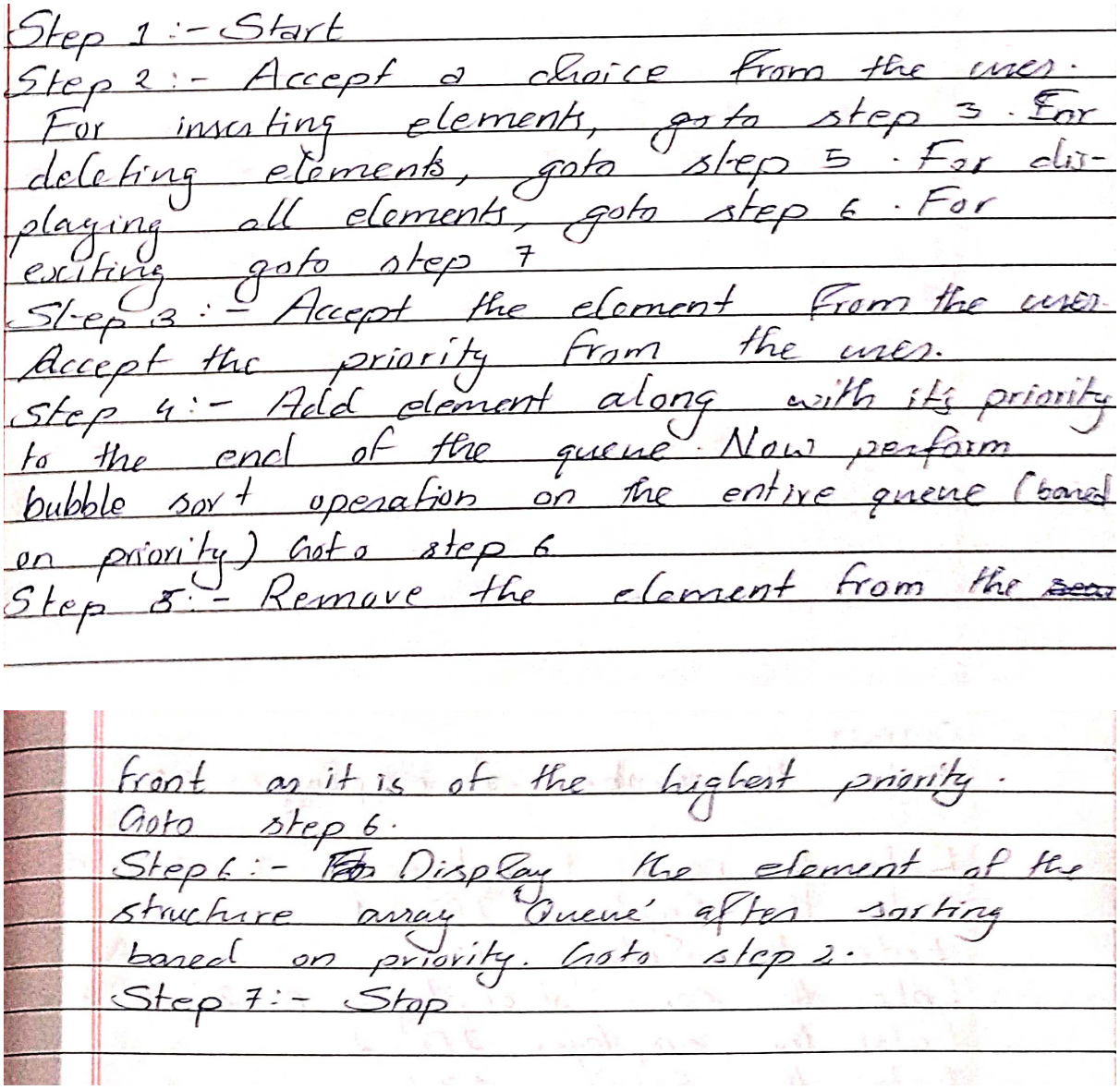
Write a menu-driven C Program to Implement a Priority Queue using arrays with the following operations:

a. Insert elements to the Priority Queue.

b. Delete elements from the Priority Queue.

c. Display the contents of the Priority Queue after each operation.

8.2 Algorithm



Step 1 :- Start
Step 2 :- Accept a choice from the user. For inserting elements, go to step 3. For deleting elements, goto step 5. For displaying all elements, goto step 6. For exiting goto step 7
Step 3 :- Accept the element from the user. Accept the priority from the user.
Step 4 :- Add element along with its priority to the end of the queue. Now perform bubble sort operation on the entire queue (based on priority) Goto step 6
Step 5 :- Remove the element from the front as it is of the highest priority. Goto step 6.
Step 6 :- Display the element of the structure array 'Queue' after sorting based on priority. Goto step 2.
Step 7 :- Stop

8.3 Code

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

#define MAX_SIZE 7

struct priorityqueue
{
    int element;
    int priority;
} priority_queue[MAX_SIZE];
int rear = -1;

void swap(int *p, int *q)
{
    int temp;
    temp = *p;
    *p = *q;
    *q = temp;
}

void sort()
{
    for (int i = 0; i <= rear; i++)
    {
        for (int j = i; j <= rear; j++)
        {
            if (priority_queue[i].priority < priority_queue[j].priority)
            {
                swap(&priority_queue[i].priority,
&priority_queue[j].priority);
                swap(&priority_queue[i].element, &priority_queue[j].element);
            }
        }
    }
}

int isEmpty()
{
    if (rear == -1)
        return 1;
    else
        return 0;
}

int isFull()
{
```

```
    if (rear == MAX_SIZE - 1)
        return 1;
    else
        return 0;
}

void Insert(int element, int priority)
{
    rear++;
    priority_queue[rear].element = element;
    priority_queue[rear].priority = priority;
}

int getHighestPriority()
{
    int priority = -1;
    if (!isEmpty())
    {
        for (int i = 0; i <= rear; i++)
        {
            if (priority_queue[i].priority > priority)
            {
                priority = priority_queue[i].priority;
            }
        }
    }
    return priority;
}

int Delete()
{
    int i, element, priority = getHighestPriority();
    for (i = 0; i <= rear; i++)
    {
        if (priority_queue[i].priority == priority)
        {
            element = priority_queue[i].element;
            break;
        }
    }


    if (i < rear)
    {
        for (int j = i; j < rear; j++)
        {
            priority_queue[j] = priority_queue[j + 1];
        }
```

```
    }
    rear--;
    return element;
}

void Display()
{
    int i;
    if (isEmpty())
    {
        printf("\nThe Queue is now empty.\n\n");
    }
    else
    {
        sort();
        printf("\nPriority Queue : \n\tFront -> ");
        for (i = 0; i <= rear; i++)
        {
            printf("%d ", priority_queue[i].element);
        }
        printf("<- Rear\n");
    }
}

void main()
{
    int response, choice, element, priority;
    do
    {
        printf("\n\nM E N U\n\n1. Enqueue elements to the Queue.\n2. Dequeue
elements from the Queue.\n3. Exit\n\n");
    ch:
        printf("\t  -> ");
        scanf("%d", &choice);
        printf("\n\n");
        switch (choice)
        {
        case 1:
            printf("Enter the element to be inserted and its priority : ");
            scanf("%d%d", &element, &priority);
            Insert(element, priority);
            Display();
            break;

        case 2:
            Delete();
            Display();
            break;
```

```
        case 3:
            exit(0);

        default:
            printf("Enter a valid Choice!!!!\n\n");
            goto ch;
        }
        printf("\nDo you want to continue (1 or 0) : ");
        scanf("%d", &response);
    } while (response == 1);
}
```

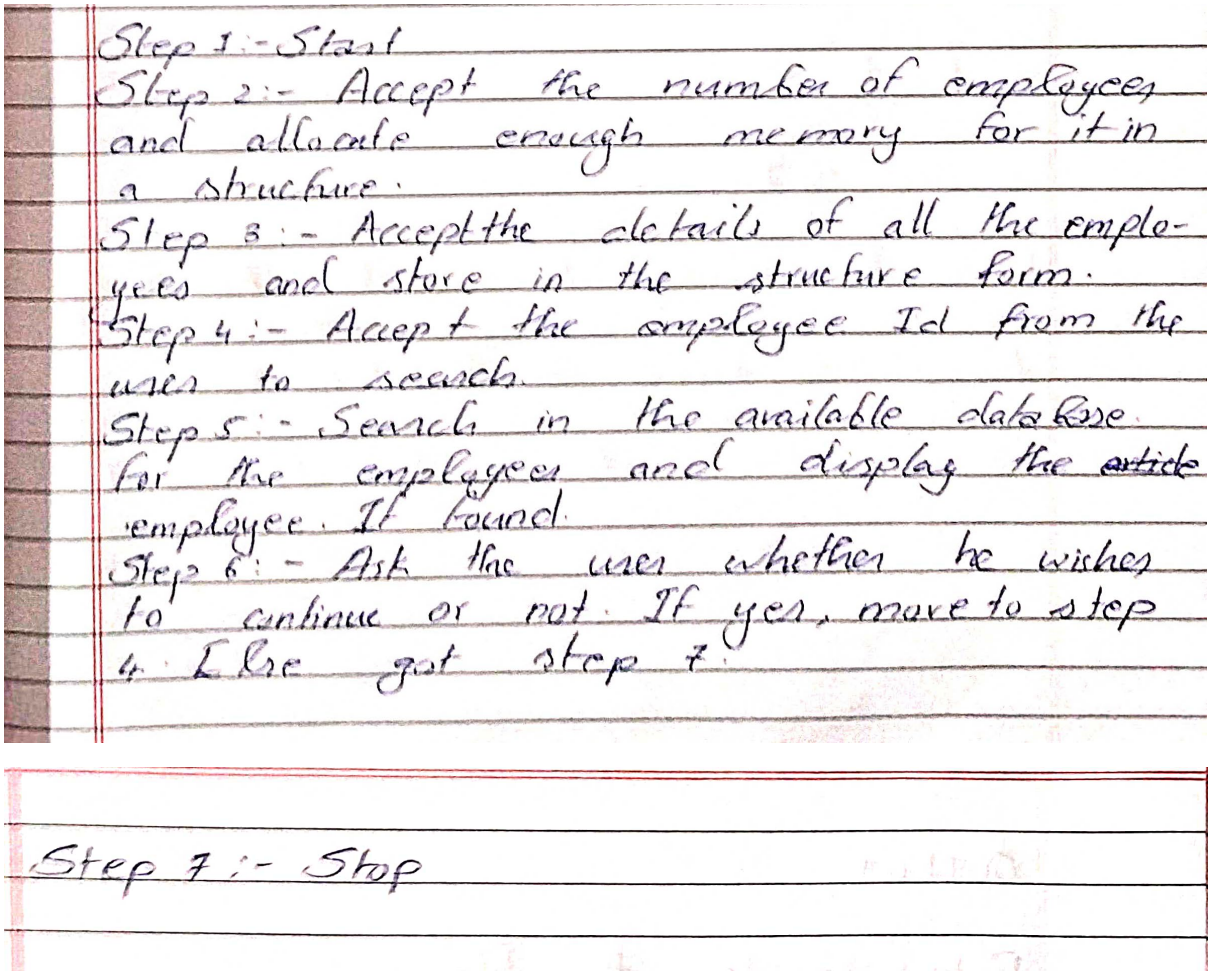
8.4 Sample Output

9 Linear Search Application

9.1 Aim

Write a C program to create a structured Employee with fields EmpId, Name, and Salary. The name should contain first name, middle name, and last name. Store the details of n employees, dynamically allocating memory for the same. Write a function to implement Linear Search to search for a particular employee, given the EmpId.

9.2 Algorithm



9.3 Code

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

typedef struct names // Defining a structure for names
{
    char First_name[20];
    char Middle_name[20];
    char Last_name[30];
```

```
} names;

typedef struct employee_details // Defining a structure for employees
{
    names Name;
    int Employee_ID;
    int Salary;
} employee_details;

int no_of_employees;
struct employee_details *details_ptr;

void Display(int i);
int search(int ID);

void main()
{
    int response, choice, employee_id;

    printf("Enter the number of employees : ");
    scanf("%d", &no_of_employees);

    details_ptr = (struct employee_details *)malloc(no_of_employees *
sizeof(employee_details));
    if (details_ptr == NULL)
    {
        printf("Error!!! memory not allocated.(details_ptr)");
        exit(0);
    }

    printf("Please enter the employee details (Name Employee_ID Salary) :
\n\n");
    for (int i = 0; i < no_of_employees; i++)
    {
        printf("Enter the name of employee no. %d (First_name Middle_name
Last_name) : \n", i + 1);
        scanf("%s %s %s", (details_ptr + i)->Name.First_name, (details_ptr +
i)->Name.Middle_name, (details_ptr + i)->Name.Last_name);
        printf("Enter the Employee ID : ");
        scanf("%d", &(details_ptr + i)->Employee_ID);
        printf("Enter the Salary : ");
        scanf("%d", &(details_ptr + i)->Salary);
    }

    do
    {
        printf("\n\n\t M E N U\n\n");
        printf("1. Search\n2. Display details of all employees\n3. Exit\n");
```

```
    ch:
        printf("\n\t->");
        scanf("%d", &choice);
        printf("\n");

        switch (choice)
        {
        case 1:
            printf("To search the employee please enter the employee ID : ");
            scanf("%d", &employee_id);
            Display(search(employee_id));
            break;

        case 2:
            for (int i = 0; i < no_of_employees; i++)
            {
                printf("\nEmployee_ID. - %d  \nName - %s %s %s \tSalary -
%d\n", (details_ptr + i)->Employee_ID, (details_ptr + i)->Name.First_name,
(details_ptr + i)->Name.Middle_name, (details_ptr + i)->Name.Last_name,
(details_ptr + i)->Salary);
            }
            break;

        case 3:
            exit(0);
            break;

        default:
            printf("\nEnter a valid Choice!!!\n");
            goto ch;
        }

        printf("\nDo you want to continue( 1 or 0 ) : ");
        scanf("%d", &response);

    } while (response == 1);
}

void Display(int i)
{
    if (i < 0)
    {
        printf("\nID not found!!!\n");
    }
    else
    {
        printf("\nEmployee_ID. - %d\nName - %s %s %s \tSalary - %d\n",
(details_ptr + i)->Employee_ID, (details_ptr + i)->Name.First_name,
```

```
(details_ptr + i)->Name.Middle_name, (details_ptr + i)->Name.Last_name,
(details_ptr + i)->Salary);
    }
}

int search(int ID)
{
    int flag, count = 0;
    for (int i = 0; i < no_of_employees; i++)
    {
        if ((details_ptr + i)->Employee_ID == ID)
        {
            flag = i;
            count++;
            break;
        }
    }
    if (count == 0)
    {
        return -1;
    }

    return flag;
}
```

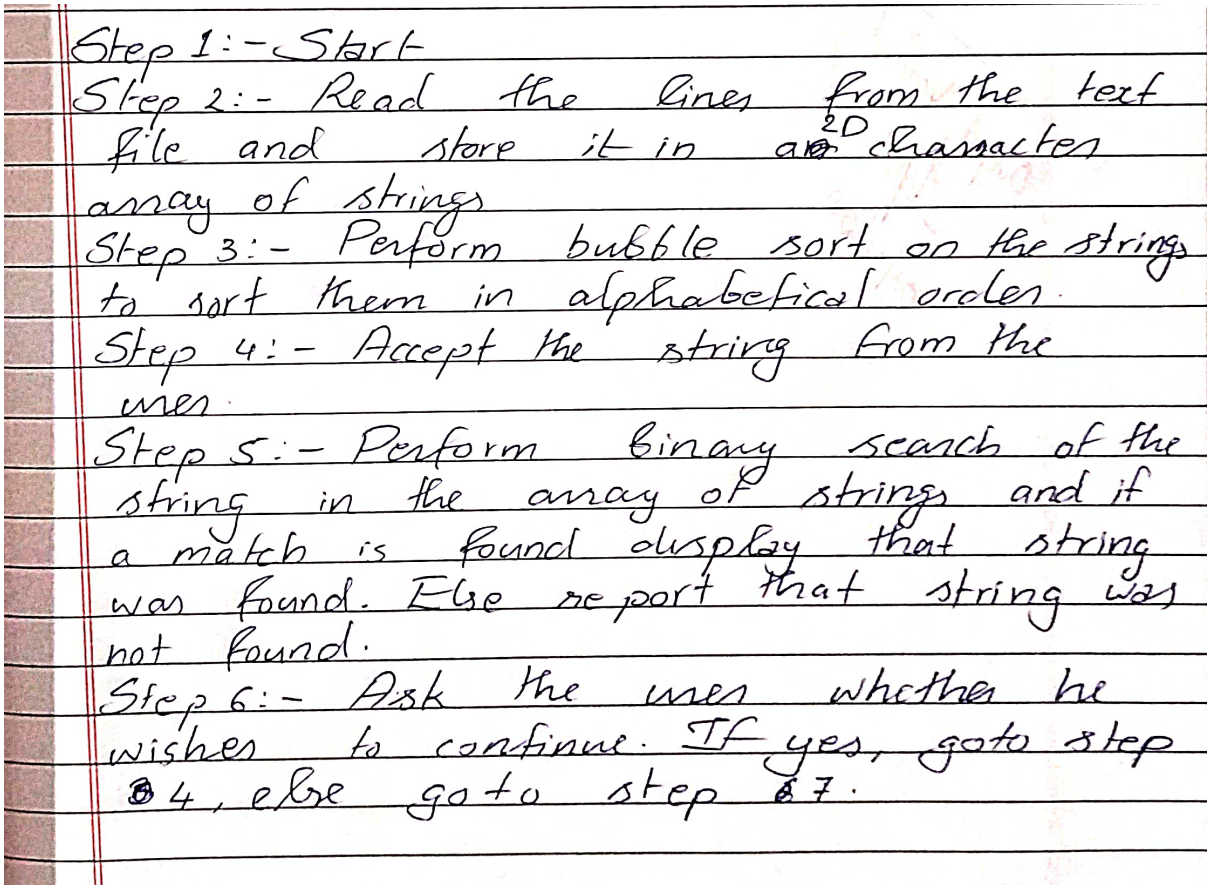
9.4 Sample Output

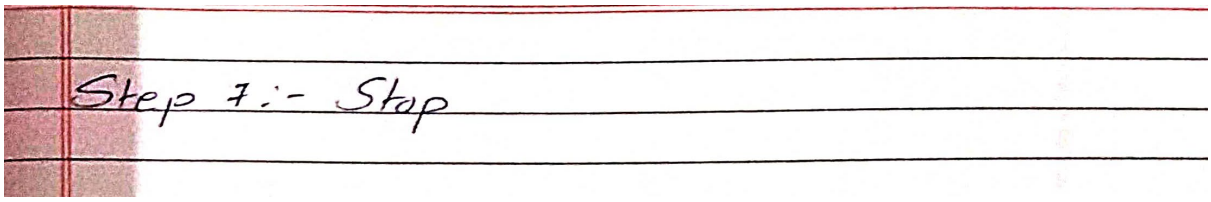
10 Binary Search Application

10.1 Aim

Write a C program to read string data stored in a file. Sort the strings in alphabetical order using Bubble Sort. Implement Binary Search to search for a given string. Implement sort and search routines as separate functions.

10.2 Algorithm





10.3 Code

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

```c
int first = 0, middle, last, no_of_words = 0;

void sort(char str[][100], int n)
{
    char temp[100];
    for (int i = 0; i < n; i++)
    {
        for (int j = i + 1; j < n; j++)
        {
            if (strcasecmp(str[i], str[j]) > 0)
            {
                strcpy(temp, str[i]);
                strcpy(str[i], str[j]);
                strcpy(str[j], temp);
            }
        }
    }
}

void binary_search(char key[100], char string[][100]);

void main()
{
    char string[10][100], a_string[100], key[100];

    FILE *file_ptr;
    file_ptr = fopen("file.txt", "r");

    if (file_ptr == NULL)
    {
        printf("\nThe file failed to open!!!\n\n");
        exit(0);
    }
    else
    {
        while (!feof(file_ptr))
        {
            fgets(a_string, 100, file_ptr);
            int len = strlen(a_string);
            if (no_of_words != 9)
            {
                a_string[len - 1] = '\0';
            }
            strcpy(string[no_of_words], a_string);
            no_of_words++;
        }
        fclose(file_ptr);
    }
```

```
    printf("Data inside the file : \n");
    for (int i = 0; i < 10; i++)
    {
        printf("%s\n", string[i]);
    }

    int response, choice;
    do
    {
        printf("\n\nM E N U\n\n1. Search\n2. Sort\n3. Exit\n\n");
    ch:
        printf("\t  -> ");
        scanf("%d", &choice);
        printf("\n\n");
        switch (choice)
        {
        case 1:
            printf("\nEnter the string to search : ");
            scanf("%s", key);
            binary_search(key, string);
            break;

        case 2:
            sort(string, no_of_words);
            for (int i = 0; i < 10; i++)
            {
                printf("%s ", string[i]);
                printf("\n");
            }
            break;

        case 3:
            exit(0);

        default:
            printf("Enter a valid Choice!!!!\n\n");
            goto ch;
        }
        printf("\nDo you want to continue (1 or 0) : ");
        scanf("%d", &response);
    } while (response == 1);
}

void binary_search(char key[100], char string[][100])
{
    last = no_of_words - 1;
    middle = (first + last) / 2;
```

```
    sort(string, no_of_words);

    while (first <= last)
    {
        if (strcasecmp(string[middle], key) < 0)
        {
            first = middle + 1;
        }
        else if (strcasecmp(string[middle], key) == 0)
        {
            printf("\n'%s' is found at location %d.\n", key, middle + 1);
            break;
        }
        else
        {
            last = middle - 1;
        }
        middle = (first + last) / 2;
    }

    if (first > last)
        printf("String not found!!! %s isn't present in the list.\n", key);
}
```

File.txt :-

Apple

Banana

Orange

Pineapple

Cherry

Kiwi

Grapes

Blueberry

Strawberry

Mango

10.4 Sample Output