1. **Develop a Program in C for the following:**
A. **Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), the second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).**
B. **Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.**

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

#define NUM_DAYS_IN_WEEK 7
// Structure to represent a day
typedef struct
{
        char *acDayName;
        int iDate;
        char *acActivity;
        } DAYTYPE;
        void fnFreeCal (DAYTYPE *);
        void fnDispCal (DAYTYPE *);
        void fnReadCal (DAYTYPE *);
        DAYTYPE *fnCreateCal();

        int main()
        {// Create the calendar
        DAYTYPE *weeklyCalendar = fnCreateCal();
        // Read data from the keyboard
        fnReadCal (weeklyCalendar);
        //display the week activity details
        fnDispCal(weeklyCalendar);
        // Free allocated memory
        fnFreeCal (weeklyCalendar);
        return 0;
}
DAYTYPE *fnCreateCal ()
{
    DAYTYPE *calendar = (DAYTYPE *)malloc( NUM_DAYS_IN_WEEK *sizeof(DAYTYPE));
   for (int i = 0; i < NUM_DAYS_IN_WEEK; i++)
    {
    calendar[i].acDayName = NULL;
    calendar[i].iDate = 0;
    calendar[i].acActivity = NULL;
    }
return calendar;
}
```

```c
void fnReadCal (DAYTYPE *calendar)
{
char cChoice;
for (int i = 0; i < NUM_DAYS_IN_WEEK; i++)
    {
    printf("Do you want to enter details for day %d [Y/N]: ", i + 1);
    scanf("%c", &cChoice);
    getchar();
    if (tolower(cChoice) == 'n')
    continue;
    printf("Day Name: ");
    char nameBuffer[50];
    scanf("%s", &nameBuffer);
     calendar[i].acDayName = strdup (nameBuffer); // Dynamically allocate and copy the string
    printf("Date: ");
    scanf("%d", &calendar[i].iDate);
    printf("Activity: ");
    char activityBuffer[100];
    scanf("%S", &activityBuffer); // Read the entire line including spaces
    calendar[i].acActivity = strdup (activityBuffer);
    printf("\n");
    getchar(); //remove trailing enter character in input buffer
    }
}
void fnDispCal (DAYTYPE *calendar)
{
printf("\nWeek's Activity Details:\n");
for (int i = 0; i < NUM_DAYS_IN_WEEK; i++)
{
printf("Day %d:\n", i + 1);
if (calendar[i].iDate == 0)
    {
    printf("No Activity\n\n");
    continue;
    }
printf(" Day Name: %s\n", calendar[i].acDayName);
printf(" Date: %d\n", calendar [i].iDate);
printf(" Activity: %s\n\n", calendar[i].acActivity);
}
}
void fnFreeCal (DAYTYPE *calendar)
{
for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
    {
    free (calendar[i].acDayName);
    free (calendar[i].acActivity);
    }
free(calendar);
}
```

OUT PUT:

o you want to enter details for day 1 [Y/N]: y	**Day 1:**
Day Name: Monday	**Day Name: Monday**
Date: 02	**Date: 02**
Activity: singing	**Activity: Singing**
Do you want to enter details for day 2 [Y/N]: y	**Day 2:**
Day Name: Tuesday	**Day Name: Tuesday**
Date: 03	**Date: 03**
Activity: Dancing	**Activity: Dancing**
Do you want to enter details for day 3 [Y/N]:y	**Day 3:**
Day Name: Wednesday	**Day Name: Wednesday**
Date: 04	**Date: 04**
Activity: Yoga	**Activity: Yoga**
Do you want to enter details for day 4 [Y/N]: n	**Day 4:**
Do you want to enter details for day 5 [Y/N]: n	**No Activity**
Do you want to enter details for day 6 [Y/N]: n	**Day 5:**
Do you want to enter details for day 7 [Y/N]: nWeek's Activity Details:	**No Activity**
	Day 6:
	No Activity
	Day 7:
	No Activity

2. Develop a Program in C for the following operations on Strings.
   A. **Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)**
   B. **Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR Support the program with functions for each of the above operations. Don't use Built-in functions.**

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
    char acMainStr[200], acSrchStr[30], acRepStr[30], acResStr[200], acCopyStr[200];
    int i=0, j=0 ,k=0, l, iMtchCnt, iStop, len, iNumOfMatch=0;

    printf("\nEnter the main string :\n");
        scanf(" %[^\n]", acMainStr);

    printf("\nEnter the Pattern string :\n");
        scanf(" %[^\n]", acSrchStr);
    printf("\nEnter the Replace string :\n");
        scanf(" %[^\n]", acRepStr);
    strcpy(acCopyStr, acMainStr);
    for(i=0;i<(strlen(acMainStr)-strlen(acSrchStr)+1);i++)
    {
        iMtchCnt = 0;
        for(j=0;j<strlen(acSrchStr);j++)
        {
            if(acMainStr[i+j] == acSrchStr[j])
            {
                iMtchCnt++;
            }
            else
            {
                break;
            }
        }
        if(iMtchCnt == strlen(acSrchStr))     //Check if number of character matches equals length of
pattern string
        {
            iNumOfMatch++;         //update number of total matches by 1
            for(k=0;k<i;k++)
            {
                acResStr[k] = acMainStr[k];       //copy till the ith character where the match occured
            }
            iStop = k + strlen(acSrchStr); //point from where rest of the original string has to be
copied
            acResStr[k] = '\0';
            strcat(acResStr, acRepStr); // append the replacement string
            len = strlen(acResStr);
```

```
                    for(k=iStop, l=0; acMainStr[k] != '\0';k++, l++) //copy rest of original string
                    {
                        acResStr[len+l] = acMainStr[k];
                    }
                    acResStr[len+l] = '\0';
                    strcpy(acMainStr,acResStr);
                }
            }
        }
    printf("\nInput Text :\n");
    printf("%s\n",acCopyStr);
    if(iNumOfMatch > 0)
    {
        printf("\n%d matches occured\n\nText after replacing matched patterns is shown below\n",
iNumOfMatch);
        printf("\n%s\n",acResStr);
    }
    else
    {
        printf("\nPattern String not found in Text\n");
    }
    return 0;
}
```

OUT PUT:

Enter the main string : venkayesh

Enter the Pattern string : y

Enter the Replace string : t

Input Text : venkayesh

1 matches occured

Text after replacing matched patterns is shown below

venkatesh

3. Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

       a. Push an Element on to Stack
       b. Pop an Element from Stack
       c. Demonstrate how Stack can be used to check Palindrome
       d. Demonstrate Overflow and Underflow situations on Stack
       e. Display the status of Stack
       f. Exit Support the program with appropriate functions for each of the above operations

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

#define MAX 4

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

int main(void)
{
int stkArray[MAX];
int top = -1;
int iElem, iChoice;
for(;;)
{
        printf("\nSTACK OPERATIONS\n");
        printf("====================");
        printf("\n 1.Push\n 2.Pop\n 3.Display\n 4.Peek\n 5.CheckPalindrome\n 6.DemonstarteOverflow\n
7.Demonstarte Underflow\n 8.EXIT\n");
        printf("Enter your choice\n");
        scanf("%d",&iChoice);
        switch(iChoice)
        {
        case 1: if(!fnStkFull(top))
           {
               printf("\nEnter element to be pushed onto the stack\n");
               scanf("%d", &iElem);
               fnPush(stkArray, iElem, &top);
           }
            else
           {
               printf("\nStack Overflow\n");
           }
break;
```

```c
        case 2: if(!fnStkEmpty(top))
            {
                iElem = fnPop(stkArray, &top);
                printf("\nPopped Element is %d\n", iElem);
            }
            else
            {
                printf("\nStack Underflow\n");
            }
break;

        case 3: if(fnStkEmpty(top))
            {
                printf("\nStack Empty\n");
            }
            else
            {
                fnDisplay(stkArray, top);
            }
                        break;

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

        case 5: printf("\nEnter number to be checked for a palindrome : ");
            scanf("%d", &iElem);
            if(fnChkPalindrome(iElem))
            {
                printf("\n%d is a palindrome\n", iElem);
            }
            else
            {
                printf("\n%d is not a palindrome\n", iElem);
            }
            break;

        case 6: if(!fnStkFull(top))
                printf("\nThere are currently %d elements in Stack\nPush %d elemnts for Stack to
overflow", top+1, MAX - (top+1));
            while(!fnStkFull(top))
            {
                printf("\nEnter an element : ");
                scanf("%d", &iElem);
```

```
            fnPush(stkArray, iElem, &top);
        }
        printf("\nStack Overflow cannot push elements onto the stack\n");
        break;

    case 7: if(!fnStkEmpty(top))
            printf("\nThere are currently %d elements in Stack\nPop out %d elemnts for Stack to
Underflow", top+1, MAX - (top+1));
        while(!fnStkEmpty(top))
        {
            iElem = fnPop(stkArray, &top);
            printf("\nPopped Element is %d\n", iElem);
        }
        printf("\nStack Underflow cannot pop elements from the stack\n");
        break;

    case 8: 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 iElem, int *t)
{
        *t = *t + 1;
        stk[*t] = iElem;
}

int fnPop(int stk[], int *t)
{
        int iElem;
        iElem = stk[*t];
        *t = *t - 1;

        return iElem;
}
```

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

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

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

bool fnChkPalindrome(int iVal)
{
    int palStk[10];
    int t = -1, iDig, iRev = 0;

    int iCopy = iVal;

    while(iCopy != 0)
    {
        iDig = iCopy % 10;
        fnPush(palStk, iDig, &t);
        iCopy /= 10;
    }
    int p = 0;
    while(p <= t)
    {
        iDig = palStk[p];
        iRev = iRev *10 + iDig;
        p++;
    }
    if(iRev == iVal)
        return true;
    else
        return false;
}
```

OUT PUT:

TACK OPERATIONS
====================
 1.Push
 2.Pop
 3.Display
 4.Peek
 5.CheckPalindrome
 6.DemonstarteOverflow
 7.Demonstarte Underflow
 8.EXIT
Enter your choice
1
Enter element to be pushed onto the stack
6
STACK OPERATIONS

TACK OPERATIONS
====================
 1.Push
 2.Pop
 3.Display
 4.Peek
 5.CheckPalindrome
 6.DemonstarteOverflow
 7.Demonstarte Underflow
 8.EXIT
Enter your choice
1
Enter element to be pushed onto the stack
5

TACK OPERATIONS
====================
 1.Push
 2.Pop
 3.Display
 4.Peek
 5.CheckPalindrome
 6.DemonstarteOverflow
 7.Demonstarte Underflow
 8.EXIT
Enter your choice
3
Stack Contents are:
        5
        6
Stack has 2 elements
STACK OPERATIONS
====================
 1.Push
 2.Pop
 3.Display
 4.Peek
 5.CheckPalindrome
 6.DemonstarteOverflow
 7.Demonstarte Underflow
 8.EXIT
Enter your choice
4
Element at the   top of the stack is 5

STACK OPERATIONS
====================
 1.Push
 2.Pop
 3.Display
 4.Peek
 5.CheckPalindrome
 6.DemonstarteOverflow
 7.Demonstarte Underflow
 8.EXIT
Enter your choice
5

Enter number to be checked for a palindrome : 1661
1661 is a palindrome

STACK OPERATIONS
====================
 1.Push
 2.Pop
 3.Display
 4.Peek
 5.CheckPalindrome
 6.DemonstarteOverflow
 7.Demonstarte Underflow
 8.EXIT
Enter your choice
8

4. Develop a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.

```c
#include <stdio.h>
#include <ctype.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(isalnum(cSymb))
{
acPost[j++] = 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))
{
if((cSymb == '^') && (acStack[top] == '^'))
break;
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;b

return elem;
}
int fnPrecd(char ch)
{
int iPrecdVal;
switch(ch)
{
case '#' : iPrecdVal = -1; break;
case '(' : iPrecdVal = 0; break;
case '+' :
case '-' : iPrecdVal = 1; break;
case '%' :
case '*' :
case '/' : iPrecdVal = 2; break;
case '^' : iPrecdVal = 3; break;
}
return iPrecdVal;
}
```

OUT PUT:

Enter a valid infix expression : (a+b)*c/d^5%1

Infix Expression is :(a+b)*c/d^5%1

Postfix Expression is :ab+c*d5^/1%

5. Develop a Program in C for the following Stack Applications
   a. **Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^**

```c
#include <stdio.h>
void push(int [], int*, int);
int pop(int [], int*);
int main()
{
int iastack[50], i, op1, op2, res;
char expr[50], symb;
int top = -1;
printf("\nEnter a valid postfix expression : \n");
scanf("%s", expr);
for(i=0; i<strlen(expr); i++)
{ symb = expr[i];
if(isdigit(symb))
{
push(iastack, &top, symb-'0');
}
else
{
op2 = pop(iastack, &top);
op1 = pop(iastack, &top);
switch(symb)
{ case '+' : res = op1 + op2;
break;
case '-' : res = op1 - op2;
break;
case '*' : res = op1 * op2;
break;
case '/' : res = op1 / op2;
break;
case '%' : res = op1 % op2;
break;
case '^' : res = (int)pow(op1 , op2);
break;
}
push(iastack, &top, res);
}
}
res = pop(iastack, &top);
printf("\nValue of %s expression is : %d\n", expr, res);
return 0;
}
void push(int Stack[], int *t , int elem)
{
*t = *t + 1;
Stack[*t] = elem;
}
```

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

OUT PUT:

Enter a valid postfix expression :
4+7*9-8+5

Value of 4+7*9-8+5 expression is : 5

5. Develop a Program in C for the following Stack Applications
b. Solving Tower of Hanoi problem with n disks

```c
#include <stdio.h>
void towers(int, char, char, char);
int main()
{
int num;
printf("Enter the number of disks : ");
scanf("%d", &num);
printf("The sequence of moves involved in the Tower of Hanoi are :\n");
towers(num, 'A', 'C', 'B');
printf("\n");
return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
if (num == 1)
{
printf("\n Move disk 1 from peg %c to peg %c", frompeg, topeg);
return;
}
towers(num - 1, frompeg, auxpeg, topeg);
printf("\n Move disk %d from peg %c to peg %c", num, frompeg, topeg);
towers(num - 1, auxpeg, topeg, frompeg);
}
```

OUT PUT:

Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from peg A to peg C
 Move disk 2 from peg A to peg B
 Move disk 1 from peg C to peg B
 Move disk 3 from peg A to peg C
 Move disk 1 from peg B to peg A
 Move disk 2 from peg B to peg C
 Move disk 1 from peg A to peg C

6. Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

       a. Insert an
       Element on to Circular QUEUE
       b. Delete an Element from Circular QUEUE
       c. Demonstrate Overflow and Underflow situations on Circular QUEUE
       d. Display the status of Circular QUEUE
       e. Exit Support the program with appropriate functions for each of the above operations.

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define SIZE 5
void insert(char [], int*, int*, char);
char del(char[], int*, int*);
void display(char [], int, int);
bool qfull(int, int);
bool qempty(int, int);
int main()
{
char q[SIZE];
int f = -1, r = -1;
int ch;
char elem;
for(;;)
{
printf("\nQueue Operations\n");
printf("=====================");
printf("\n1.Qinsert\n2.Qdelete\n3.Qdisplay\n4.Exit\n");
printf("Enter your choice\n");
scanf("%d",&ch);
getchar();
switch(ch)
{
case 1: if(!qfull(f,r))
{
printf("\nEnter an element : ");
scanf("%c", &elem);
insert(q, &f, &r, elem);
}
else
{
printf("\nQueue is Full\n");
}
break;

case 2: if(!qempty(f, r))
```

```c
{
elem = del(q, &f, &r);
printf("\nDeleted element is %c\n", elem);
}
else
{
printf("\nQueue is Empty\n");
}
break;
case 3: if(!qempty(f, r))
{
printf("\nContents of the Queue is \n");
display(q, f, r);
}
else
{
printf("\nQueue is Empty\n");
}
break;
case 4: exit(0);
default: printf("\nInvalid choice\n");
break;
}
}
return 0;
}
bool qfull(int fr, int rr)
{
if((rr+1) % SIZE == fr)
return true;
else
return false;
}
bool qempty(int fr, int rr)
{
if(fr == -1)
return true;
else
return false;
}
void insert(char queue[], int *f, int *r, char val)
{
if(*r == -1)
{
*f = *f + 1;

*r = *r + 1;
}
else
```

```c
*r = (*r + 1)%SIZE;
queue[*r] = val;
}
char del(char queue[], int *f, int *r)
{
char el;
el = queue[*f];
if(*f == *r)
{
*f = -1;
*r = -1;
}
else
{
*f = (*f + 1)%SIZE;
}
return el;
}
void display(char queue[], int fr, int rr)
{
int i;
if(fr<=rr)
{
for(i=fr; i<=rr; i++)
{
printf("%c\t", queue[i]);
}
printf("\n");
}
else
{
for(i=fr; i<=SIZE-1; i++)
{
printf("%c\t", queue[i]);
}
for(i=0; i<=rr; i++)
{
printf("%c\t", queue[i]);
}
printf("\n");
}
}
```

OUT PUT:

```
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
1
Enter an element : 6
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
1
Enter an element : 8
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
Invalid choice
Queue Operations
==================
```

```
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
2
Deleted element is
5
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
3
Contents of the Queue is
5      8

Queue
Operations====================
```

```
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
1
Enter an element : 5
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
Invalid choice
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
3
Contents of the Queue is
6      5      8
Queue Operations
====================
1.Qinsert
```

```
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
3
Contents of the Queue is
8
Queue Operations
====================
1.Qinsert
2.Qdelete
3.Qdisplay
4.Exit
Enter your choice
3
Queue is Empty
```

7. Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo
  a. Create a SLL of N Students Data by using front insertion.
  b. Display the status of SLL and count the number of nodes in it
  c. Perform Insertion / Deletion at End of SLL
  d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
  e. Exit

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
char usn[11], name[40], prog[4];
int sem;
char ph[11];
struct node *link;
};
typedef struct node* PTR;
PTR get(void);
void freeN(PTR);
PTR insrear(PTR);
PTR delfront(PTR);
PTR insfront(PTR);
PTR delrear(PTR);
void disp(PTR);
int main()
{
PTR first = NULL;
int ch, num, i;
printf("\nEnter the number of Students N : ");
scanf("%d", &num);
for(i=0;i<num;i++)
{
printf("\nEnter Data for Node %d :\n", i+1);
first = insfront(first);
}
for(;;)
{
printf("\nQUEUE OPERATIONS\n");
printf("====================");
printf("\n1.Insert Front\n2.Insert Rear\n3.Delete Front\n4.Delete Rear\n5.Display\n6.Exit\n");
printf("\nEnter your choice\n");
scanf("%d",&ch);

switch(ch)
{
case 1: first = insfront(first);
break;
```

```
case 2: first = insrear(first);
break;
case 3: first = delfront(first);
break;
case 4: first = delrear(first);
break;
case 5: disp(first);
break;
case 6: exit(0);
}
}
return 0;
}
PTR get()
{
PTR newborn;
newborn = (PTR)malloc(sizeof(struct node));
if(newborn == NULL)
{
printf("\nMemory Overflow");
exit(0);
}
printf("\nEnter USN : ");
scanf("%s",newborn->usn);
printf("\nEnter name : ");
scanf("%s",newborn->name);
printf("\nEnter Program name : ");
scanf("%s", newborn->prog);
printf("\nEnter semester : ");
scanf("%d",&newborn->sem);
printf("\nEnter Phone no : ");
scanf("%s",newborn->ph);
return newborn;
}
void freeN(PTR x)
{
free(x);
}

PTR insrear(PTR first)
{
PTR temp,cur;
temp = get();
temp->link = NULL;
if(first == NULL)
return temp;
cur = first;
while(cur->link != NULL)
{
```

```
cur = cur->link;
}
cur->link = temp;
return first;
}
PTR delfront(PTR first)
{
PTR temp;
if(first == NULL)
{
printf("\nSLL is empty cannot delete\n");
return first;
}
temp = first;
first = first->link;
printf("\nNode deleted is %s\n",temp->name);
freeN(temp);
return first;
}
void disp(PTR first)
{
PTR curr;
int count = 0;
if(first == NULL)
{
printf("\nSLL is empty\n");
return;
}
printf("\nThe contents of SLL are :\n");
curr = first;
printf("\nUSN\t\tName\tProgram\tSem\tPhone num");
while(curr != NULL)
{
printf("\n%10s\t%s\t%s\t%d\t%s",curr->usn, curr->name, curr->prog, curr->sem, curr->ph);

curr = curr->link;
count++;
}
printf("\n\nSLL has %d nodes\n", count);
}
PTR insfront(PTR first)
{
PTR temp;
temp = get();
temp->link = NULL;
temp->link = first;
first = temp;
return first;
}
```

```
PTR delrear(PTR first)
{
PTR cur, prev;
if(first == NULL)
{
printf("\nSLL is empty cannot delete\n");
return first;
}
prev = NULL;
cur = first;
if(cur->link == NULL)
{
printf("\nNode deleted for %s\n",cur->name);
freeN(cur);
return NULL;
}
while(cur->link != NULL)
{
prev = cur;
cur = cur->link;
}
prev->link = cur->link;
printf("\nNode deleted for %s\n",cur->name);
freeN(cur);
return first;
}
```

OUT PUT:

Enter the number of Students N : 2	Enter USN : 2VX22CB064
Enter Data for Node 1 :	Enter name : Amit
Enter USN : 2VX22CB063	Enter Program name : CSBS
Enter name : Venkatesh	Enter semester : 3
Enter Program name : CSBS	Enter Phone no : 1234567890
Enter semester : 3	QUEUE OPERATIONS
Enter Phone no : 8296360743	
Enter Data for Node 2 :	
QUEUE OPERATIONS	QUEUE OPERATIONS
====================	====================
1.Insert Front	1.Insert Front
2.Insert Rear	2.Insert Rear
3.Delete Front	3.Delete Front
4.Delete Rear	4.Delete Rear

5.Display 6.Exit Enter your choice 1 Enter USN : 2VX22CB063 Enter name : Venkatesh Enter Program name : CSBS Enter semester : 3 Enter Phone no : 8596360743	5.Display 6.Exit Enter your choice 2 Enter USN : 2VX22CB064 Enter name : Amit Enter Program name : CSBS Enter semester : 3 Enter Phone no : 1234567890

QUEUE OPERATIONS **====================** **1.Insert Front** **2.Insert Rear** **3.Delete Front** **4.Delete Rear** **5.Display** **6.Exit** **Enter your choice** **3** **Node deleted is Venkatesh** **QUEUE OPERATIONS**	**QUEUE OPERATIONS** **====================** **1.Insert Front** **2.Insert Rear** **3.Delete Front** **4.Delete Rear** **5.Display** **6.Exit** **Enter your choice** **3** **Node deleted is Amit**

QUEUE OPERATIONS **====================** **1.Insert Front** **2.Insert Rear** **3.Delete Front** **4.Delete Rear** **5.Display** **6.Exit** **Enter your choice** **4** **Node deleted for Amit** **QUEUE OPERATIONS**	**QUEUE OPERATIONS** **====================** **1.Insert Front** **2.Insert Rear** **3.Delete Front** **4.Delete Rear** **5.Display** **6.Exit** **Enter your choice** **5** **The contents of SLL are :**

	USN Name Program Sem Phone num
	2VX22CB063 Venkatesh CSBS 3 8296360743
	SLL has 1 nodes

8. Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo
   a. Create a DLL of N Employees Data by using end insertion.
   b. Display the status of DLL and count the number of nodes in it
   c. Perform Insertion and Deletion at End of DLL
   d. Perform Insertion and Deletion at Front of DLL
   e. Demonstrate how this DLL can be used as Double Ended Queue.
   f. Exit

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node
{
int usn;
char name[30], dept[4], desig[30], ph[11];
int sal;
struct node *plink;
struct node *nlink;
};
typedef struct node* NODE;
NODE getn(void);
void freen(NODE);
NODE insrear(NODE);
NODE delfront(NODE);
NODE insfront(NODE);
NODE delrear(NODE);
void disp(NODE);
int main()
{
NODE first = NULL;
int ch, num, i;
printf("\nEnter the number of Employees N : "); scanf("%d", &num);
for(i=0;i<num;i++)
{
printf("\nEnter Data for Node %d :\n", i+1);
first = insrear(first);
}
for(;;)
{
printf("\nDLL OPERATIONS\n");
printf("===================");

printf("\n1.Insert Rear\n2.Delete Front\n3.Insert Front\n4.Delete Rear\n5.Display\n6.Exit\n");
printf("\nEnter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1: first = insrear(first);
```

```
break;
case 2: first = delfront(first);
break;
case 3: first = insfront(first);
break;
case 4: first = delrear(first);
break;
case 5: disp(first);
break;
case 6: exit(0);
}
}
return 0;
}
NODE getn()
{
NODE newborn;
newborn = (NODE)malloc(sizeof(struct node));
if(newborn == NULL)
{
printf("\nMemory Overflow");
exit(0);
}
printf("\nEnter SSN : ");
scanf("%d",&newborn->usn);
printf("\nEnter name : ");
scanf("%s",newborn->name);
printf("\nEnter Department : ");
scanf("%s", newborn->dept);
printf("\nEnter Designation : ");
scanf("%s", newborn->desig);
printf("\nEnter Salary : ");
scanf("%d",&newborn->sal);
printf("\nEnter Phone no : ");
scanf("%s",newborn->ph);
return newborn;
}

void freen(NODE x)
{
free(x);
}
NODE insrear(NODE first)
{
NODE temp,cur;
temp = getn();
temp->plink = temp->nlink = NULL;
if(first == NULL)
return temp;
```

```
cur = first;
while(cur->nlink != NULL)
{
cur = cur->nlink;
}
cur->nlink = temp;
temp->plink = cur;
return first;
}
NODE insfront(NODE first)
{
NODE temp;
temp = getn();
temp->plink = temp->nlink = NULL;
temp->nlink = first;
first = temp;
return first;
}
NODE delrear(NODE first)
{
NODE cur, prev;
if(first == NULL)
{
printf("\nDLL is empty\n");
return first;
}

cur = first;
if(cur->nlink == NULL)
{
printf("\nNode deleted for %s\n",cur->name);
freen(cur);
return NULL;
}
while(cur->nlink != NULL)
{
cur = cur->nlink;
}
prev = cur->plink;
prev->nlink = NULL;
printf("\nNode deleted for %s\n",cur->name);
freen(cur);
return first;
}
NODE delfront(NODE first)
{
NODE temp;
if(first == NULL)
{
```

```
printf("\nDLL is empty\n");
return first;
}
if(first->nlink == NULL)
{
printf("\nNode deleted for %s\n",first->name);
freen(first);
return NULL;
}
temp = first;
first = first->nlink;
first->plink = NULL;
printf("\nNode deleted for %s\n",temp->name);
freen(temp);
return first;
}

void disp(NODE first)
{
NODE curr;
int count = 0;
if(first == NULL)
{
printf("\nDLL is empty\n");
return;
}
printf("\nThe contents of DLL are :\n");
curr = first;
printf("\nSSN\tName\tDept\tDesignation\tSalary\t\tPhone No");
while(curr != NULL)
{
printf("\n%-5d\t%s\t%s\t%s\t\t%-7d\t\t%-11s",curr->usn, curr->name, curr->dept, curr->desig,
curr->sal, curr->ph);
curr = curr->nlink;
count++;
}
printf("\n\nDLL has %d nodes\n", count);
}
```

OUT PUT:

Enter the number of Employees N : 1	Enter Department : CSE
Enter Data for Node 1 :	Enter Designation : aim
Enter SSN : 321	Enter Salary : 20000

Enter name : xyz

Enter Department : CSE

Enter Designation : aim

Enter Salary : 10000

Enter Phone no : 8296360743

DLL OPERATIONS
=====================
1.Insert Rear
2.Delete Front
3.Insert Front
4.Delete Rear
5.Display
6.Exit

Enter your choice
1

Enter SSN : 322

Enter name : abc

Enter Phone no : 1234567890

DLL OPERATIONS
=====================
1.Insert Rear
2.Delete Front
3.Insert Front
4.Delete Rear
5.Display
6.Exit

Enter your choice
5

The contents of DLL are :

SSN   Name   Dept   Designation
Salary   Phone No
321   xyz   CSE   aim   10000
8296360743
322   abc   CSE   aim   20000
1234567890

DLL has 2 nodes

DLL OPERATIONS
=====================
1.Insert Rear
2.Delete Front
3.Insert Front
4.Delete Rear
5.Display
6.Exit

Enter your choice
3

Enter SSN : 963

Enter name : ven

Enter Department : CSE

Enter Designation : aim

Enter Salary : 50000

DLL OPERATIONS
=====================
1.Insert Rear
2.Delete Front
3.Insert Front
4.Delete Rear
5.Display
6.Exit

Enter your choice
4

Node deleted for abc

DLL OPERATIONS
=====================
1.Insert Rear
2.Delete Front
3.Insert Front
4.Delete Rear
5.Display
6.Exit

Enter Phone no : 963852741

DLL OPERATIONS

Enter your choice
2

Node deleted for ven

DLL OPERATIONS
====================
1.Insert Rear
2.Delete Front
3.Insert Front
4.Delete Rear
5.Display
6.Exit

Enter your choice
5

The contents of DLL are :

SSN	Name	Dept	Designation	Salary	Phone No
321	XYZ	CSE	aim	10000	8296360743

DLL has 1 nodes

9. Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes

     a. Represent and Evaluate a Polynomial
       $P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3$
     b. Find the sum of two polynomials POLY1(x,y,z)
       and POLY2(x,y,z) and store the result in POLYSUM(x,y,z) Support the
    program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <math.h>
struct polyt
{
int cf,px, py,pz;
struct polyt* next;
};
typedef struct polyt* PTR;
PTR insert(PTR poly, int cf, int px, int py, int pz)
{
PTR cur;
PTR nn = (PTR)malloc(sizeof(struct polyt));
nn->cf = cf;
nn->px = px;
nn->py = py;
nn->pz = pz;
nn->next = NULL;
cur = poly;
while(cur->next != poly)
{
cur = cur->next;
}
cur->next = nn;
nn->next = poly;
return poly;
}

void disp(PTR poly)
{
if (poly->next == poly)
{
printf("Polynomial is empty.\n");
return;
}
PTR cur = poly->next;
do
{
printf("%dx^%dy^%dz^%d ", cur->cf, cur->px, cur->py, cur->pz);
```

```
cur = cur->next;
if (cur != poly)
{
printf("+ ");
}
} while (cur != poly);
printf("\n");
}
int evaluate(PTR poly, int x, int y, int z)
{
int result = 0;
if (poly->next == poly)
{
return result;
}
PTR cur = poly->next;
do
{
int termValue = cur->cf;
termValue *= pow(x, cur->px);
termValue *= pow(y, cur->py);
termValue *= pow(z, cur->pz);
result += termValue;
cur = cur->next;
} while (cur != poly);
return result;
}

bool fmatch(PTR p1, PTR p2)
{
bool match = true;
if(p1->px != p2->px)
match = false;
if(p1->py != p2->py)
match = false;
if(p1->pz != p2->pz)
match = false;
return match;
}
PTR add(PTR poly1, PTR poly2, PTR polySum)
{
PTR cur1 = poly1->next;
PTR cur2 = poly2->next;
do
{
polySum = insert(polySum, cur1->cf, cur1->px, cur1->py, cur1->pz);
cur1 = cur1->next;
} while(cur1 != poly1);
do
```

```
{
cur1 = polySum->next;
bool matchfound = false;
do
{
if(fmatch(cur1, cur2))
{
cur1->cf += cur2->cf;
matchfound = true;
break;
}
cur1 = cur1->next;
} while(cur1 != polySum);
if(!matchfound)
{
polySum = insert(polySum, cur2->cf, cur2->px, cur2->py, cur2->pz);
}
cur2 = cur2->next;
} while(cur2 != poly2);
return polySum;
}

int main()
{
PTR poly1 = (PTR)malloc(sizeof(struct polyt));
poly1->next = poly1;
PTR poly2 = (PTR)malloc(sizeof(struct polyt));
poly2->next = poly2;
PTR polySum = (PTR)malloc(sizeof(struct polyt));
polySum->next = polySum;
poly1 = insert(poly1, 6, 2, 2, 1);
poly1 = insert(poly1, 4, 0, 1, 5);
poly1 = insert(poly1, 3, 3, 1, 1);
poly1 = insert(poly1, 2, 1, 5, 1);
poly1 = insert(poly1, 2, 1, 1, 3);
// Display the polynomial P(x, y, z)
printf("POLY1(x, y, z) = ");
disp(poly1);
// Read and evaluate the second polynomial POLY2(x, y, z)
// Represent the polynomial P(x, y, z) = xyz + 4x^3yz
poly2 = insert(poly2, 1, 1, 1, 1); // Example term
poly2 = insert(poly2, 4, 3, 1, 1);
// Display the second polynomial POLY2(x, y, z)
printf("POLY2(x, y, z) = ");
disp(poly2);
// Add POLY1(x, y, z) and POLY2(x, y, z) and store the result in POLYSUM(x, y, z)
polySum = add(poly1, poly2, polySum);
// Display the sum POLYSUM(x, y, z)
printf("\nPOLYSUM(x, y, z) = ");
```

```
disp(polySum);
// Evaluate POLYSUM(x, y, z) for specific values
int x = 1, y = 2, z = 3;
int res = evaluate(polySum, x, y, z);
printf("\nResult of POLYSUM(%d, %d, %d): %d\n", x, y, z, res);
return 0;
}
```

OUT PUT :

POLY1(x, y, z) = 6x^2y^2z^1 + 4x^0y^1z^5 + 3x^3y^1z^1 + 2x^1y^5z^1 + 2x^1y^1z^3
POLY2(x, y, z) = 1x^1y^1z^1 + 4x^3y^1z^1

POLYSUM(x, y, z) = 6x^2y^2z^1 + 4x^0y^1z^5 + 7x^3y^1z^1 + 2x^1y^5z^1 + 2x^1y^1z^3 + 1x^1y^1z^1

Result of POLYSUM(1, 2, 3): 2364

10. Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers . a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2 b. Traverse the BST in Inorder, Preorder and Post Order c. Search the BST for a given element (KEY) and report the appropriate message d. Exit

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
int info;
struct node *lbranch;
struct node *rbranch;
};
typedef struct node* NODEPTR;
NODEPTR fnGetNode(void);
void fnFreeNode(NODEPTR x);
NODEPTR fnInsertNode(int, NODEPTR);
void fnInOrder(NODEPTR);
void fnPreOrder(NODEPTR);
void fnPostOrder(NODEPTR);
void fnSearchBST(NODEPTR, int);
int main()
{
NODEPTR root = NULL;
int iChoice, iItem, i, iNum;
printf("Create a BST of N Integers \n");
printf("\nEnter the number N : ");
scanf("%d", &iNum);
printf("\nEnter %d numbers\n", iNum);
for(i=0;i<iNum;i++)
{
scanf("%d", &iItem);
root = fnInsertNode(iItem,root);
}
for(;;)
{
printf("\n1.Inorder traversal\n2.Preorder traversal");
printf("\n3.Postorder traversal\n4.Search\n5.Exit\n");printf("\nEnter your choice : ");
scanf("%d",&iChoice);
switch(iChoice)
{
case 1: if(root ==NULL)
{
```

```c
printf("\nTree is Empty\n");
}
else
{
printf("\nInorder Traversal is :\n");
fnInOrder(root);
printf("\n");
}
break;
case 2: if(root ==NULL)
{
printf("\nTree is Empty\n");
}
else
{
printf("\nPreorder Traversal is :\n");
fnPreOrder(root);
printf("\n");
}
break;
case 3: if(root ==NULL)
{
printf("\nTree is Empty\n");
}
else
{
printf("\nPostorder Traversal is :\n");
fnPostOrder(root);
printf("\n");
}
break;
case 4: printf("\nEnter the element to be searched : ");
                              scanf("%d", &iItem);

fnSearchBST(root, iItem);
break;
case 5: exit(0);
default: printf("Wrong choice\n");
break;
}
}
return 0;
```

```
}
NODEPTR fnGetNode(void)
{
NODEPTR x;
x = ( NODEPTR ) malloc (sizeof(struct node));
if(x == NULL)
{
printf("\nOut of Memory");
exit(0);
}
return x;
}
void fnFreeNode(NODEPTR x)
{
free(x);
}
NODEPTR fnInsertNode(int iItem,NODEPTR root)
{
NODEPTR temp,prev,cur;
temp = fnGetNode();
temp->info = iItem;
temp->lbranch = NULL;
temp->rbranch = NULL;
if(root == NULL)
                                return temp;

prev = NULL;
cur = root;
while(cur != NULL)
{
prev = cur;
if(iItem == cur->info)
{
printf("\nDuplicate items not allowed\n");
fnFreeNode(temp);
return root;
}
cur = (iItem < cur->info)? cur->lbranch: cur->rbranch;
}
if(iItem < prev->info)
prev->lbranch = temp;
else
prev->rbranch = temp;
```

```
return root;
}
void fnPreOrder(NODEPTR root)
{
if(root != NULL)
{
printf("%d\t",root->info);
fnPreOrder(root->lbranch);
fnPreOrder(root->rbranch);
}
}
void fnInOrder(NODEPTR root)
{
if(root != NULL)
{
                              fnInOrder(root->lbranch);

printf("%d\t",root->info);
fnInOrder(root->rbranch);
}
}
void fnPostOrder(NODEPTR root)
{
if(root != NULL)
{
fnPostOrder(root->lbranch);
fnPostOrder(root->rbranch);
printf("%d\t",root->info);
}
}
void fnSearchBST(NODEPTR root, int iElem)
{
if(root != NULL)
{
if(iElem < root->info)
fnSearchBST(root->lbranch, iElem);
else if(iElem > root->info)
fnSearchBST(root->rbranch, iElem);
else
printf("\n%d is found in the BST\n",iElem);
}
else
```

```
{
printf("\n%d is not found in the BST\n",iElem);
}
                                            }
```

OUT PUT :

Create a BST of N Integers	1.Inorder traversal
Enter the number N : 10	2.Preorder traversal
Enter 10 numbers	3.Postorder traversal
2020	4.Search
3050	5.Exit
2103	
2546	Enter your choice : 1
23546	
2453	Inorder Traversal is :
6589	2020 2103 2453 2546 3050 3254
3254	6541 6589 7896 23546
7896	
6541	
1.Inorder traversal	1.Inorder traversal
2.Preorder traversal	2.Preorder traversal
3.Postorder traversal	3.Postorder traversal
4.Search	4.Search
5.Exit	5.Exit
Enter your choice : 2	Enter your choice : 3

Preorder Traversal is :						Postorder Traversal is :					
2020	3050	2103	2546	2453	23546	2453	2546	2103	6541	3254	7896
6589	3254	6541	7896			6589	23546	3050	2020		