1.

To develop a c program that fulfills the specified requirements, including declaring a calender as an array of 7 elements, dynamically allocating memory for strings and implementing the create(), read(), and display() functions, you can use the following co

```
#include <stdio.h>
#include <stdlib.h>
#define MALLOC(p, n, type)\
{\
p = (type *) malloc (n * sizeof(type));\
if (p == NULL)\
{\
printf("Insufficient Memory\n");\
exit(0);\
}\
}
typedef struct {
    char *dayName;
    char *date;
    char *activity;
} Day;
void create (Day *calendar) {
  int i;
  for (i = 0; i < 7; i++) {
     MALLOC (calender[i].dayName, 20, char);
     MALLOC (calender[i].date, 20, char);
     MALLOC (calendar[i].activity, 100, char);
  }
}
```

```
void read(Day *calendar) {
    int i;
    for(i=0; i<7; i++) {
    fprintf("Enter the name of the day for day %d:",
                i+1);
    scanf("%s", calendar[i].dayName);
    printf("Enter the day for day %d:", i+1);
    scanf("%s", calendar[i].date);
    fprintf("Enter the activity description for day %d:");
    scanf("%[^\n]s", calendar[i].activity);
    }
}

void display(Day *calendar) {
    int i;
    printf("Weekly Activity Report:\n");
    for(i=0; i<7; i++) {
        fprintf("Day %d:\n", i+1);
        fprintf("DayName: %s\n", calendar[i].dayName);
        fprintf("Date: %s\n", calendar[i].date);
        fprintf("Activity: %s\n", calendar[i].activity);
        fprintf("\n");
    }
}

int main() {
    Day *calendar;
    int i;
    MALLOC(calendar, 7, Day);
    create(calendar);
    read(calender);
    display(calendar);
    for(i=0; i<7; i++) {
        free(calendar[i].dayName);
        free(calendar[i].activity);
    }
    return 0;
}
```

8/12/2023

2. Develop a C program for the following operation in a
a. Read a main string (STR), a pattern string (PAT) and replace string (REP).
b. Perform pattern matching operation : Find and replace all occurances of PAT in str with REP if PAT exists in STR. ~~Report a~~ Report suitable messag in case PAT does not exits in STR.
Support the program with function for each of t above equations. Dont use Built-in functions.

```
#include <stdio.h>
char str[30], pat[20], Rep[50], Res[100];
int i=0; j=0; c=0; k=0; m=0; flag=0;
void stringmatch()
{
  while (str[c]!='\0')
  {
    if (str[m]==pat[i])
    {
      m++;
      i++;
      if (pat[i]=='\0')
      {
        flag=1;
        for (k=0; Rep[k]!='\0', k++, j++)
        {
          Res[j] = Rep[k];
        }
        i=0;
        c=m;
      }
    }
  }
```

```
        else
        {
            Res[j] = str[c]
            j++;
            c++;
            i = 0;
            m = c;
        }
    }
    Res[j] = '\0';
}
void main()
{
    printf("Enter the main string\n");
    gets(str);
    printf("Enter the pattern\n");
    gets(pat);
    printf("Enter the replace string\n");
    gets(rep);
    printf("\n the string before pattern matching %s", str);
    stringmatch()
    if (flag==1)
    printf("Patter matched and it is replaced %s", rep)
    else
        printf("pattern not found\n");
}
```

O/P: Enter the main string
mit ise
Enter the Pattern
ise
Enter the replace string
mys
The string before the patter matching is mit ise
Patter matched and is replaced mit mys

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.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 3
int stop = -1;
void push (int item);
int pop ();
void palindrome ();
void display ();
void main () {
int choice, item;
while (1) {
printf ("\n\n\n\n ~~~ menu ~~~ :");
printf ("\n => 1. push an element the stack and
        Overflow demo");
printf ("\n => 2. pop an element from stack and
        Underflow demo");
printf ("\n => 3. Palindrome Demo");
printf ("\n => 4. Display");
printf ("\n => 5. Exit");
printf ("\n Enter your choice \n");
```

```
scanf("%d", &choice);
switch(choice) {
case 1:
    printf("\n Enter an element to be pushed:\n");
    scanf("%d", &item);
    push(item);
    break;
case 2:
    item = pop();
    if (item != -1)
        printf("Element poped is: %d\n", item);
    break;
case 3:
    palindrome();
    break;
case 4:
    display();
    break;
case 5:
    exit(1);
default:
    printf("Please enter valid choice\n");
    break;
  }
 }
}
void push(int item) {
if (top == MAX-1) {
   printf("\n ~~~ Stack overflow ~~~");
   return;
}
top = top+1;
```

```
    s[top] = item;
}
int pop() {
    int item;
    if (top == -1) {
        printf("\n ~~~ Stack underflow ~~~");
        return -1;
    }
    item = s[top];
    top = top - 1;
    return item;
}
void display() {
    int i;
    if (top == -1) {
        printf("\n ~~~ Stack is empty ~~~");
        return;
    }
    printf("\n Stack elements are : \n");
    for (i = top; i > 0; i--) {
        printf("%d\n", s[i]);
}
void palindrome() {
    int flag = 1;
    printf("\n Stack content are : \n");
    for (i = top; i >= 0; i--)
        printf("%d\n", s[i]);
    printf("\n Reverse of Stack content are : \n");
    for (i = 0; i <= top; i++) {
        printf("%d\n", s[i]);
    for (i = 0; i <= top/2; i++) {
        if (s[i] != s[top - i]) {
```

```
        flag = 0;
        break;
      }
    }
    if (flag == 1) {
        printf("\n it is palindrome number");
    } else {
        printf("\n it is not palindrome number");
    }
}
```

symbols	Stack precedence function (F)	Input precedence function (G)	Associative
+, -	2	1	left
*, -	4	3	left
$ or ^	5	6	right
operands	8	7	left
(	0	9	left
)	-	0	-
#	-1	-	-

(A + (B - C) * D)

Stack	s[top]	symbol	F(s[top]) > G(symbol)	postfix
#	#	(	-1 < 9 (push '(')	-
#(	(	A	0 < 7 push 'A'	-
#(A	A	+	8 > 1 pop A & print	A
	(	+	0 < 1 push +	A
#(+	+	(	2 > 9 push (	A
#(+(	(	B	0 < 7 push B	AB
#(+(B	B	-	8 > 1 pop and print B	AB
#(+(	(	-	0 > 1 push -	AB
#(+(-	-	C	2 < 7 push C	AB
#(+(-C	C	)	8 > 0 pop & print C	ABC
	-	)	2 > 0 pop and print -	ABC-
	(	)	0 = 0 pop & print (	ABC-

ACHIEVER
Date : / /

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 +, -, *, /, % (Reminder) ^ (Power) and alphanumeric operands.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int F(char symbol)
{
  switch(symbol){
  case '+':
  case '-': return 2;
  case '*':
  case '%':
  case '/': return 4;
  case '^':
  case '$': return 5;
  case '(': return 0;
  case '#': return -1;
  default : return 8.
  }
}
int G(char symbol)
{
  switch(symbol)
  {
    case '+':
    case '-': return 1;
    case '%': return 2;
    case '/': return 3;
    case '^':
```

```
        case '$' : return 6;
        case '(' : return 9;
        case ')' : return 0;
        default : return 7;
    }
}

void infix_postfix (char infix[], char postfix[])
{
    int top, i, j;
    char s[i];
    char symbol;
    top = -1;
    s[++top] = '#';
    j = 0;
    for (i = 0; i < strlen(infix); i++)
    {
        symbol = infix[i];
        while (F(s[top]) > G(symbol))
            postfix[j++] = s[top--];
        if (F(s[top]) != G(symbol))
        {
            s[++top] = symbol;
        else
            top--;
        }
    }
    while (s[top] != '#') {
        postfix[j++] = s[top--];
        postfix[j] = '\0';
    }
}
void main()
{
    char infix[20], postfix[20];
```

```
printf("enter the valid infix expression\n");
scanf("%s", infix);
infix_postfix(infix, postfix);
printf("the postfix expression is \n");
printf("%s\n", postfix);
}
```

1. O/P :- Calendar:

Enter the day name for the day 1: Monday
Enter the date for the day 1: 01.12.2023
Enter the activity desprription for the day 1: Partying
Enter the dayname for the day 2: Tuesday.
Enter the date for the day 2: 02.12.2023
Enter the activity description for the day 2: Sleeping
Enter the day name for the day 3: Wednesday
Enter the date for the day 3: 03.12.2023
Enter the activity description for the day 3: Working
Enter the day name for the day 4: Thursday
Enter the date for the day 4: 04.12.2023
Enter the activity description for the day 4: Writing
Enter the day name for the day 5: Friday
Enter the date for the day 4: 05.12.2023
Enter the activity description for the day 5: Studying.
Enter the day name for the day 6: Saturday
Enter the date for the day 6: 06.

ACHIEVER
Date : / /

5b. Develop a C program for the stack application of solving Towers of Hanoi problem with n disks.

```
#include <stdio.h>
#  void tower (int n, int source, int temp, int destination)
{
    if (n == 0)
    return;
    tower(n-1, source, destination, temp);
    printf ("\n Move disc %d from %c to %c", n, source,
            destination);
    tower(n-1, temp, source, destination);
}
void main()
{
    int n;
    printf ("\n Enter the number of discs: \n");
    scanf ("%d", &n);
    tower(n, 'A', 'B', 'C');
    printf("\n\n Total Number of moves are : %d", (int)
            pow (2, n) -1);
}
```

→ O/P:

Move disc 1 from A to C
Move disc 2 from A to B
Move disc 1 from C to B
Move disc 3 from A to C
Move disc 1 from B to A
Move disc 2 from B to C
Move disc 1 from A to C.
Total numbers of moves are: 7

5a) Develop a program in C for the following stack applications, evaluation of suffix expression with single digit operands : +, -, *, /, %, ^

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int top = -1, i;
int op1, op2, res, s[20];
char postfix[20], symbol;
void push (int item)
{
    top = top + 1;
    s[top] = item;
}
int pop() {
    int item;
    item = s[top];
    top = top - 1;
    return item;
}
void main() {
    printf("Enter the postfix =\n");
    scanf("%s", postfix);
    for (i=0; postfix[i] != '\0', i++)
    {
        symb = postfix[i];
        if (isdigit(symb)) {
            push(symb - '0');
        }
        else
        {
```

```
op2 = pop();
op1 = pop();
switch (symb) {
    case '+': push (op1 + op2)
              break;
    case '-': push (op1 - op2)
              break;
    case '*': push (op1 * op2);
              break;
    case '/': push (op1 / op2);
              break;
    case '%': push (op1 % op2);
              break;
    case '$': push (op1 $ op2);
              break;
    case '^': push (pow (op1, op2));
              break;
    default : push (0);
        }
    }
}

res = pop();
printf ("result = %d\n", res);
}
```

→ O/P:

i. enter the postfix expression
123+*321-+*
result = 20.

ii. Enter the postfix expression
632-5*+1$7+
result = 18.

iii. Enter the postfix expression
12+3-21+3$-
result = -27

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
d. Display the status of circular queue
e. Exit.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
char q[SIZE];
int i, r = -1;
int option, f = 0;
int j, count = 0;
void insert() {
    if (count == SIZE)
    {
        printf ("Queue is Full \n");
        return;
    }
    else
    {
        r = (r+1) % QUEUE;
        printf ("Enter the item \n");
        scanf ("%c", &q[r]);
        count ++;
    }
}
void delete()
{
```

```
    if (count == 0)
    {
      printf("Queue is empty\n");
      return;
    }
    else
    {
      printf("Deleted item is %c", q[f]);
      count--;
      f = (f+1) % SIZE;
    }
}
void display()
{
  if (count == 0)
  {
    printf("Queue is empty\n");
    return;
  }
  else
  {
    if i = f;
    for(j=1; j<=count; j++)
    {
      printf("%c", q[i]);
      i = (i+1) % SIZE;
    }
  }
}
int main()
{
  for(;;)
```

```
}
printf("1. Insert, 2. Delete \n 3. Display, 4. Exit \n");
printf("Enter your option \n");
scanf("%d", &option);
switch(option)
{
  case 1: insert();
          break;
  case 2: delete();
          break;
  case 3: display();
          break;
  case 4: exit(0);
          break;
  default: printf("Invalid choice");
          break;
  }
 }
}
```

O/P.

1. Insert
2. Delete
3. Display
4. Exit

Enter your choice

1

Enter the element to the queue

2

1. Insert
2. Delete
3. Display
4. Exit

Enter your choice

1