Program-01

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.

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

typedef struct
 {
  char *dayName;
  int date;
  char *activity;
 }week;

void create(week *day)
 {
  day->dayName = (char *)malloc(sizeof(char) * 20);
  day->activity = (char *)malloc(sizeof(char) * 100);

  printf("Enter the day name: ");
  scanf("%s", day->dayName);

  printf("Enter the date: ");
  scanf("%d", &day->date);

  printf("Enter the activity for the day: ");
  scanf(" %[^\n]s", day->activity);
 }

void read(week *calendar, int size)
 { int i;
  for (i = 0; i < size; i++) {
      printf("Enter details for Day %d:\n", i + 1);
      create(&calendar[i]);
  }
 }
void display(week *calendar, int size)
 {
  int i;
  printf("\nWeek's Activity Details:\n");
  printf("---------------------------------------------------\n");
printf("Dayno\t\tDayname\t\tDate\t\tActivity\n");
printf("---------------------------------------------------\n");
for (i = 0; i < size; i++)
  {
      printf("%d\t\t", i + 1);
      printf("| %s |\t\t", calendar[i].dayName);
      printf("| %d |\t\t", calendar[i].date);
      printf("| %s |\t\t", calendar[i].activity);
      printf("\n");
  }
 }

int main() {
  int size;
  week *calendar;
  printf("Enter the number of days in the week: ");
scanf("%d", &size);

  calendar = (week *)malloc(sizeof(week) * size);

  if (calendar == NULL) {
      printf("Memory allocation failed. Exiting program.\n");
      return 1;
  }

  read(calendar, size);
  display(calendar, size);

  free(calendar);

  return 0;
 }
```

Program-02

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. `#include<stdio.h>`

```
char str[50], pat[20], rep[20], res[50];
int c = 0, m = 0, i = 0, j = 0, k, flag = 0;

void stringmatch()
{
 while (str[c] != '\0')
 {
 if (str[m] == pat[i])
 {
```

```
 i++;
 m++;
 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++;
 m = c;
 i = 0;
 }
 }
 res[j] = '\0';
}
void main()
{
 printf("Enter the main string:");
 gets(str);
 printf("\nEnter the pat string:");
 gets(pat);
 printf("\nEnter the replace string:");
 gets(rep);
 printf("\nThe string before pattern match is:\n %s", str);
stringmatch();
 if (flag == 1)
 printf("\nThe string after pattern match and replace is: %s ",  res);
 else
 printf("\nPattern string is not found");
}
```

Program-03

Design, Develop and Implement 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 operation.

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

#define MAX 4

int s[MAX];
int top = -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 to 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("\nEnter your choice: ");
  scanf("%d", & choice);
  switch (choice)
  {
       case 1: printf("\nEnter an element to be pushed: ");
                      scanf("%d", & item);
            push(item);
  break;
  case 2: item = pop();
  if (item != -1)
  printf("\nElement popped is: %d", item);  break;
  case 3: palindrome();
            break;
  case 4: display();
  break;
  case 5: exit(1);
  default: printf("\nPlease enter valid choice ");  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("\nStack elements are:\n ");
  for (i = top; i >= 0; i--)
  printf("| %d |\n", s[i]);
 }
void palindrome()
 {
  int flag = 1, i;
  printf("\nStack content are:\n");
  for (i = top; i >= 0; i--)
  printf("| %d |\n", s[i]);

  printf("\nReverse 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("\nIt is palindrome number");  }
  else
  {
  printf("\nIt is not a palindrome number");  }
 }
```

Program -04

Design, Develop and Implement 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.

ABOUT THE EXPERIMENT:
Infix: Operators are written in-between their operands. Ex: X + Y
Prefix: Operators are written before their operands. Ex: +X Y
Postfix: Operators are written after their operands. Ex: XY+
Examples of Infix, Prefix, and Postfix
Infix Expression Prefix Expression Postfix Expression
A + B + A B A B +
A + B * C + A * B C A B C * +

Infix to prefix conversion
Expression = **(A+B^C)*D+E^5**
Step 1. Reverse the infix expression.
5^E+D*)C^B+A(
Step 2. Make Every '(' as ')' and every ')' as '('
5^E+D*(C^B+A)
Step 3. Convert expression to postfix form. A+(B*C-(D/E-F)*G)*H

Expression Stack Output Comment 5^E+D*(C^B+A) Empty - Initial ^E+D*(C^B+A) Empty 5 Print E+D*(C^B+A) ^ 5 Push +D*(C^B+A) ^ 5E Push D*(C^B+A) + 5E^ Pop And Push *(C^B+A) + 5E^D Print (C^B+A) +* 5E^D Push C^B+A) +*(5E^D Push ^B+A) +*(5E^DC Print B+A) +*(^ 5E^DC Push +A) +*(^ 5E^DCB Print A) +*(+ 5E^DCB^ Pop And Push) +*(+ 5E^DCB^A Print End +* 5E^DCB^A+ Pop Until '(' End Empty 5E^DCB^A+*+ Pop Every element

Step 4. Reverse the expression.
+*+A^BCD^E
Step 5. Result
+*+A^BCD^E5

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

int F(char symbol)
{
       switch (symbol)
       {
        case '+':
        case '-':return 2;
        case '*':
        case '/':
        case '%':return 4;
        case '^':
        case '$':return 5;
        case '(':return 0;
        case '#':return -1;
        default :return 8;
       }
}
int G(char symbol)
{
       switch (symbol)
       {
        case '+':
        case '-':return 1;
        case '*':
        case '/':
        case '%':return 3;
        case '^':
        case '$':return 6;
        case '(':return 3;
        case ')':return 0;
```

```
                default :return 7;
              }
}
void infix_postfix(char infix[], char postfix[])
{
        int top=-1, j=0, i;
        char s[30], symbol;
        s[++top] = '#';
        for(i=0; i < strlen(infix); i++)
        {
                symbol = infix[i];
                while (F(s[top]) > G(symbol))
                {
                        postfix[j] = s[top--];
                        j++;
                }
                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("\nEnter a valid infix expression\n") ;
      scanf ("%s", infix) ;
      infix_postfix (infix, postfix);
      printf("\nThe infix expression
      is:\n"); printf ("%s",infix);
      printf("\nThe postfix expression is:\n");
      printf ("%s",postfix) ;
}
```

PROGRAM -05 A

Design, Develop and Implement a Program in C for the following Stack Application A)

Evaluation of Suffix expression with single-digit operands and operators:+, -, *, /, %, ^

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

int i, top = -1;
int op1, op2, res, s[20];
char postfix[90], symb;

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("\nEnter a valid postfix expression:\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 '$':
        case '^': push(pow(op1, op2));
        break;
 default: push(0);
 }
 }
 }
 res = pop();
 printf("\n Result = %d", res);
}
```

PROGRAM -05 B

Design, Develop and Implement a Program in C for the following Stack Application. b. Solving Tower 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("\nMove disc %d from %c to %c", n, source, destination);
tower(n - 1, temp, source, destination);
 }

void main()
 {
  int n;
  printf("\nEnter the number of discs: \n");
  scanf("%d", & n);
  tower(n, 'A', 'B', 'C');
  printf("\n\nTotal Number of moves are: %d", (int) pow(2, n) - 1); }
```