01/01/2024

8) write a program to simulate the working of stack using an array with the following a) push b) pop c) display.

```
void push(int item)
{
  if (top == n-1)
  {
    printf("Overflow");
  }
  else
  {
    top = top+1;
    stack[top] = item;
    printf("value %od has been pushed", item);
  }
}
void pop
{
  if (top == -1)
  {
    printf("Underflow");
  }
  else
  {
    item = stack[top];
    top--;
  }
}
```

```
void display()
{
  if (top == -1)
  printf("Stack is empty");
  }
  else
    for (int i=top; i>=0; i--)
    {
      printf("%d \n", stack[top]);
    }
}

void main()
{ int choice;
  printf(" 1. PUSH  2. POP  3. DISPLAY
        4. EXIT");
printf("Enter the choice is 1,2,3,4");
scanf("%d", &choice);
    switch (choice)
  {
    case 1: printf("Enter element to be pushed");
            scanf("%d \n", item);
            break;
    case 2: pop();
            break;
    case 3: display();
            break;
    default : exit(0);
    }
  }
}
```

Q. Write a program to convert infix to postfix expression. The expression consists of single character operands and the binary operators +(plus), -(minus), *(multiply) and /(divide)

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
int count=0, pos=0, top=-1, len;
char symb, temp;
void push (char symb)
{
  top = top+1;
  stack[top] = symb;
}
char pop()
{
  symb = stack[top];
  top = top-1;
  return symb;
}
int preced (char symb)
{
  int p;
  switch symb)
  {
  case '^': p=3;
            break;
  case '/':
  case '*': p=2;
            break;
  case '-':
  case '+': p=1;
            break;
```

```
case '(':
case ')': p=0;
return p;
}

void itop (char infix[], char postfix[])
{
   char symbol, temp;
   len = strlen(infix);
   while(count <len)
   {
   symbol = infix[count];
   switch (symbol)
   {
   case '(': push(symbol);
             break;
   case ')': temp = pop();
             while (temp != '(')
             {
              postfix[pos] = temp;
               pos++;
               temp = pop();
             }
             break;

   case '+':
   case '-';
   case '/';
   case '*': while (precedstack[top] >= preced
                                    (symbol))
             {temp = pop();
              postfix[pos] = temp;
               pos++;
             }
```

```
        push(symbol);
        break;
default: postfix[post++] = symbol;
        break;
    }
    count++;
  }
}
void main()
{
  char infix, postfix;
  printf("Enter infix expression");
  scanf("%s", infix);
  inp(infix, postfix);
  printf("infix exp = %s\n", infix);
  printf("Postfix exp = %s\n", postfix);
}
```

01/01/24

Output:

Enter infix expression
(4+(3/7)-(6+8)+2/8-4
437/68+-28/+4-

Enter infix expression:
a+b*c-d+h*i

abc*+d-hi*+