store
67 8|10|24

Q) infix to postfix using operand +, -, *, /, (, )

```
int index=0, pos=0, top=-1, length;
char symbol, temp, infix[20], posfix[20], stack[20];

Void infixtopostfix();
Void push (char symbol);
 char pop();
 int pred (char symb);

Void main()
{
 clrscr();
 pf ("enter infix expression: \n");
 sf ("%s" infix)
 infix to postfix()
 pf (infix expression is)
 pf (postfix expression is)
 getch()
}

Void ~~main~~() infixtopostfix()
 {
  length = strlen (infix);
  push ('#');
  while (index < length)
  { symbol - infix [index];
  switch (symbol) {
   case '(' : push (symbol);
    breake;
   case ')':
   ~~push (symbol);~~ temp = pop();
   ~~breake;~~
```

```
while (temp != 'c' )
{
  postfix [pos++] = temp ;
  poss++;
  temp = pop();
  }
break;
case '+':
case '-':
case '*':
case '/':
case '^': while (pred (stack [top ] >= pred (symbol ))
case '(' {
        temp= pop();
        postfix [pos++ ]= temp;
        }
        push (symbol );
        break;
  default : postfix [pos ++ ]=symbol;
  }
index ++;
 }
```

```
while (top > 0)
{
temp = pop();
postfix [pos++] = temp
}
}
void push (char symbol)
{
 top = top + 1;
  stack[top] = symbol;
}
char symb;
return stack [top--];
}
int pred (char symbol)
{
int p;
switch (symbol)
{
 case '^':


 case '*';


 case '/':


 case '+':


case '^':
  while (pred (stack[top]) >= pr (symbol)){
```

```
        temp = pop();
        postfix[pos++] = temp;
      }
    push(symbol);
    break;
    default:
      postfix[pos++] = symbol;
    }
    index++;
  }
while (top > 0) {
    temp = pop(); }
    postfix[pos] = '\0';
}

char pop() {
    char symb = stack[top];
    top--;
    return symb;
}
int pre (char symb) {
  int p
  switch (symbol) {
  case '^':
      p = 3;
    break;
  case '*':
  case '/':
      p = 2;
    break;
```

```
case 'c':
    p=0;
    break;
 case '#'
    p=-1;
    break;
}
return p;
}
```

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

int index=0, pos=0, top=-1, length;
char symbol, temp, infix[30], postfix[30], stack[30];

Void infixTopostfix();
Void push (char symbol);
 char pop();
 int precedence(char symb);
int main(){
   printf("Enter infix expression:\n");
   scanf("%s", &infix);
 infixTOpostfix();
 printf("\n infix expression:\n %s", infix);
 printf("\n postfix expression:\n %s", postfix);
  return 0;
}
Void infixTopostfix(){
  length = strlen(infix)
   push = ('#');
  while (index < length){
      symbol = infix[index];
        switch (symbol){
   Case '(':
        push (symbol);
         break;
   Case ')':
        temp = pop();
        while (temp != '('){
          postfix[pos++]=temp;
          temp=pop();
        }
```

```
break;
case '+':
case '-':
case '*':
case '/':
case '^':
 while(precedence(stack[top]) >= precedence(symbol)){
    temp= pop();
    postfix[pos++] = temp;
 }
 push(symbol);
 break;
default:
    postfix[pos++] = temp;
 }
 postfix[pos] = '\0';
}
void push(char symbol){
    top=top+1;
    stack[top] = symbol;

char pop(){
    char symb = stack[top];
     top--;
     return symb;
}
```

```
int precedence(char symbol){
int p;
 switch(symbol){
     case '^':
        p = 3;
        break;
     case '*':
     case '/':
        p = 2;
        break;
     case '+':
     case '-':
        p = 1;
        break;
     case '(':
        p = 0;
        break;
     case '#':
        p = -1
        break;
   }
  return p;
}
```

Output

```
enter infix expression:

a^bc-d+e/f*(g+h)

postfix code
abc^d-eb/gh+/+.
```