```
//INFIX TO POSTFIX
#include<stdio.h>
#include<string.h>
char tos(char stack[8], int *topPtr){                    //function to return top of stack
        if(*topPtr==-1)
                return '\0';
        else
                return stack[*topPtr];
}
int precedence(char oprtr){                                        //function to determine the
preference of the operator
        switch(oprtr){
                case '+':
                        return 1;
                break;
                case '-':
                        return 1;
                break;
                case '*':
                        return 2;
                break;
                case '/':
                        return 2;
                break;
                case '\0':
                        return 0;
                break;
        }
}
char pop(char stack[8], int *topPtr){                              //function to pop the element
        return stack[*topPtr];
}
void push(char stack[8], char value, int *topPtr){          //function to push the element
        stack[++*topPtr] = value;
}
int main(){
        char infixStack[10], operatorStack[8];
        int top=-1, i=0, size;
        printf("ENTER THE INFIX EQUATION: ");
        scanf("%s", infixStack);
        printf("HERE IS THE POSTFIX EQUATION: ");
        size = strlen(infixStack);

        while(i<size){
                if(( infixStack[i] >= 97 && infixStack[i] <= 122 ) || ( infixStack[i] >= 65 && infixStack[i] <=
90 )){
                        printf("%c", infixStack[i]);
                        i++;
```

```
                }else{
                        if(precedence(infixStack[i]) <= precedence(tos(operatorStack, &top))){
                                printf("%c",pop(operatorStack, &top));
                                top--;
                                push(operatorStack, infixStack[i], &top);
                        }else{
                                push(operatorStack, infixStack[i], &top);
                        }
                        i++;
                }
        }
                while(top!=-1){
                        printf("%c",pop(operatorStack, &top));
                        top--;
                }
        return 0;
}
```

```
#include<stdio.h>
int main(){
int stack[5], optn, top =-1, i;
while(optn!=5){
printf("WHICH OPERATION DO YOU WANT TO PERFORM \n1 POP \n2 PUSH \n3 TOS \n4 TRAVERSE \n5
EXIT\n------> ");
scanf("%d", &optn);
switch(optn){
   case 1:
      if(top==-1)
         printf("STACK UNDERFLOW\n\n");
      else{
        printf("%d POPPED\n\n", stack[top]);
        top--;
      }
   break;
   case 2:
      if(top==4)
         printf("STACK OVERFLOW\n\n");
      else{
      top++;
      printf("ENTER DATA: ");
      scanf("%d", &stack[top]);
      printf("%d PUSHED\n\n", stack[top]);
      }
   break;
   case 3:
         if(top==-1)
                printf("STACK EMPTY\n\n");
```

```
            else
            printf("TOS = %d\n\n", stack[top]);
    break;
    case 4:
    if(top!=-1){
            printf("HERE IS YOUR STACK\n");
            for(i=top; i>=0; i--){
            printf("\t%d\n", stack[i]);
            }
    }else
            printf("STACK EMPTY");
    printf("\n\n");
    break;
    case 5:
       printf("EXITED SUCCESSFULLY\n\n");
    break;
    default:
       printf("ENTER VALID CHOICE\n\n");
}
}
return 0;
}
```

```
//INFIX TO PREFIX
#include<stdio.h>
#include<string.h>
void prtStr(char str[10]){
            printf("\n---->%s\n", str);
}
char tos(char stack[8], int *topPtr){                    //function to return top of stack
            if(*topPtr==-1)
                        return '\0';
            else
                        return stack[*topPtr];
}
int precedence(char oprtr){                                          //function to determine the
preference of the operator
            switch(oprtr){
                        case '+':
                                    return 1;
                        break;
                        case '-':
                                    return 1;
                        break;
                        case '*':
                                    return 2;
                        break;
                        case '/':
                                    return 2;
```

```
                break;
                case '\0':
                        return 0;
                break;
        }
}
char pop(char stack[8], int *topPtr){                                   //function to pop the element
        return stack[*topPtr];
}
void push(char stack[8], char value, int *topPtr){          //function to push the element
        stack[++*topPtr] = value;
}
int main(){
        char infixStack[10], operatorStack[8], reversedStack[10];
        int top=-1, i=0, size, top1 = -1, top2;
        printf("ENTER THE INFIX EQUATION: ");
        scanf("%s", infixStack);
        size = strlen(infixStack);
        top2 = size-1;

        //REVERSING STACK;
        while(top2!=-1){
                push(reversedStack, pop(infixStack, &top2), &top1);
                top2--;
        }

        //logic for changine from infix to prefix
        while(i<size){
                if(( reversedStack[i] >= 97 && reversedStack[i] <= 122 ) || ( reversedStack[i] >= 65 &&
reversedStack[i] <= 90 )){
                        push(infixStack, reversedStack[i], &top2);
                        top1--;
                        i++;
                }else{
                        if(precedence(reversedStack[i]) < precedence(tos(operatorStack, &top))){
                                push(infixStack, (pop(operatorStack, &top)), &top2);
                                top--;
                                push(operatorStack, reversedStack[i], &top);
                        }else{
                                push(operatorStack, reversedStack[i], &top);
                        }
                        i++;
                }
        }
        while(top!=-1){
                        push(infixStack, pop(operatorStack, &top), &top2);
                        top--;
        }
```

```
        //REVERSING STACK;
        top1 = -1;
        while(top2!=-1){
                push(reversedStack, pop(infixStack, &top2), &top1);
                top2--;
        }

        printf("HERE IS THE PREFIX EQUATION: ");
        printf("%s", reversedStack);
        return 0;
}
```

```
#include<stdio.h>

void increase(int *lnPtr, int *valPtr, int *prevNumPtr){
   *lnPtr += *valPtr;
   *prevNumPtr = *lnPtr;
   printf("%d", *lnPtr);
}

int main()
{
int i=0, j=0, nextDec = 1, prevNum=1, largeNum = 1;
char sequence[10];
printf("ENTER SEQUENCE(ALL CHARACTERS IN CAPITAL): ");
scanf("%s", sequence);
printf("HERE IS THE SHORTEST NUMBER FOR THE GIVEN SEQUESNCE: ");
if(sequence[i]=='I'){
   printf("1");
}else{
   while(sequence[j]=='D'){
      j++;
   }
  printf("%d", j+1);
}
while(sequence[i]!='\0'){
   nextDec = 1;
   if(sequence[i]=='I'){
      j = i+1;
      while(1){
      if(sequence[j]=='D')
         nextDec++;
      else
         break;
      j++;
      }
   increase(&largeNum, &nextDec, &prevNum);
   }
```

```
   else{
      if(prevNum>1)
      printf("%d", --prevNum);
      else{
      j = i+1;
      nextDec++;
         while(1){
            if(sequence[j]=='D')
               nextDec++;
            else
               break;
            j++;
         }

         largeNum = prevNum = nextDec;
         printf("%d", --prevNum);
      }
   }
    i++;
}
   return 0;
}
```