PROLOG ACADEMY

- **DATA STRUCTURE**

By- MASEERA ALI

- Book followed - Data structures by Seymour Lipschutz (Schaum Series)

LET’S START!

Reverse a string using stack

- 1) Create an empty stack.
- 2) One by one push all characters of string to stack.
- 3) One by one pop all characters from stack and put them back to string.

Code to reverse a string

```
#include <stdio.h>
#include <string.h>
#define MAX 20
int top=-1;
char stack [MAX];
char pop( )
{
if (top ==-1)
printf ("stack underflow \n");
else
return stack [top--];
}
void push(char item)
{
if (top==MAX-1)
printf ("Stack overflow\n");
else
stack[++top] = item;
return;
}
int main ( )
{

char str[20];
int i;
printf ("\nEnter the string :\n");
gets (str);
for (i=0; i<strlen(str); i++)
push (str[i]);
for (i=0; i<strlen(str); i++)
str[i]=pop ( );
printf ("\nReversed string is :\n");
puts (str);

}
```

Infix to prefix conversion

Expression = (A+B^C)*D+E^5

Step 1. Read the infix expression from the right.

Step 2. Apply the algo of conversion of infix to postfix form.

5E^DCB^A+*+

Step 3. Reverse the final expression to get your prefix expression

+*+A^BCD^E5

Infix to prefix conversion

```
#include<stdio.h>
#include<string.h>
#define MAX 20

char stack[MAX];
int top=-1;
void push (char item)
{
top++;
stack[top]=item;
return;
}

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

```
int prcd(char symbol)
   {
        switch(symbol)
        {
        case '+':
        case '-':
        return 2;
        case '*':
        case '/':
        return 3;
        case '^':
        return 4;
        case '(':
        case ')':
        return 1;
        }
    return 0;
    }
```

```
int isoperator (char symbol)
    {
        switch (symbol)
        {
        case '+':
        case '-':
        case '*':
        case '/':
        case '^':
        case '(':
        case ')':
        return 1;
        default:
        return 0;
        }
   }


 void convertip(char infix[ ], char prefix [ ])
 {
         int i, symbol, j=0;
         stack[++top]=')';
         for(i=strlen(infix)-1;i>=0;i--)
                 {
                 symbol=infix[i];
                 if(isoperator(symbol)==0)
                         {
                         prefix[j]=symbol;
                         j++;
                         }
                 else
                   {
                         if(symbol==')')
                          {
                                 push(symbol);
                           }
                         else if(symbol=='(')
                         {
                                 while (stack[top]!=')')
                                 {
                                    prefix[j]=pop( );
                                    j++;
                                 }
                         pop ();
                         }
                         else
                         {
                                 while(prcd(symbol)<=prcd(stack[top]))
                                         {
                                         prefix[j]=pop( );
                                         j++;
                                         }
                                 push(symbol);
                         }
                   }
   }
```

```
  while(stack[top]!=')')
        {
        prefix[j]=pop( );
        j++;
        }
        prefix[j]='\0';
        for (i=0; i<strlen(prefix); i++)
        push (prefix[i]);
        for (i=0; i<strlen(prefix); i++)
        prefix[i]=pop ( );
        return;
}
int main ()
        {
        char infix[20], prefix[20];
        printf("\nEnter the valid infix string:\n");
        gets(infix);
        convertip(infix, prefix);
        printf("The corresponding prefix string is:\n");
        puts(prefix);
        return 0;
        }
```

Decimal to Binary

```
#include <stdio.h>
int main ( )
{
int stack[30], dec, rem, top=0;
printf ("\nEnter decimal number:\n");
scanf("%d", &dec);
while (dec!=0)
{
rem=dec%2;
top++;
stack[top]=rem;
dec=dec/2;
}
printf ("\nThe equivalent binary number is\n");
for (; top>0; top--)
{
printf ("%d", stack[top]);
}
printf("\n");
}
```