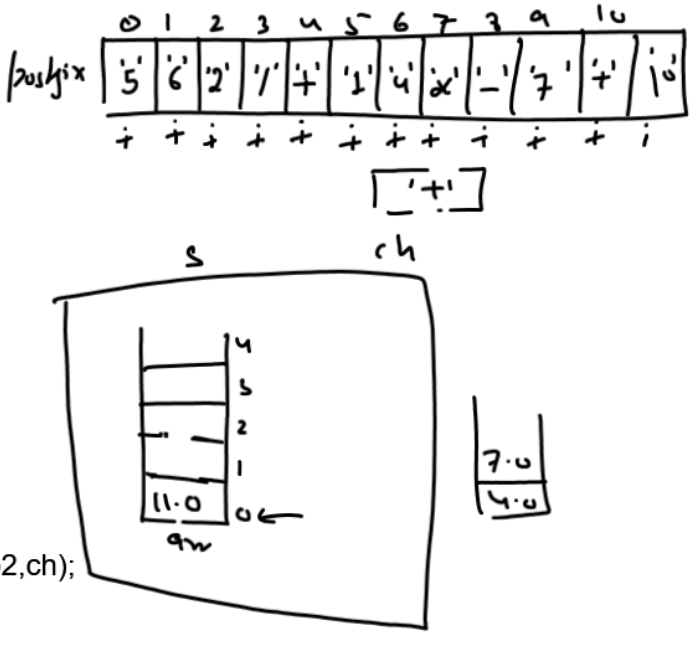
Evaluating A Postfix Expression

```
struct Stack
{
   float arr[10];
   int tos;
};
void push(struct Stack *,float);
float pop(struct Stack *);
int isoperand(char);
float calculate(float,float,char);
float evaluate(char [ ]);
int main()
{
   char postfix[20];
   float ans;
   printf("Enter a valid postfix exp:");
   scanf("%s",postfix);
   ans=evaluate(postfix);
   printf("Ans is %f",ans);
   return 0;
}
```

```
float evaluate(char postfix[20])
{
   struct Stack S;
   int i;
   char ch;
   float op1,op2,res;
   S.tos=-1;
   for(i=0;postfix[i]!='\0';i++)
   {
      ch=postfix[i];
      if(isoperand(ch)==1)
      {
         push(&s,ch-48);
       }
      else
      {
            op2=pop(&S);
            op1=pop(&S);
            res=calculate(op1,op2,ch);
            push(&S,res);
      }
  }
res=pop(&S);
return res;
}
```



```
void push(struct Stack *P,float x)
{
   if(P->tos==9)
     {
          printf("Stack Overflow");
          return;
     }
P->tos++;
P->arr[P->tos]=x;
}
float pop(struct Stack *P)
{
   float x;
   if(P->tos==-1)
   {
         printf("Stack underflow");
         return -1.0;
   }
   x=P->arr[P->tos];
   P->tos--;
   return x;
}
```

```
int isoperand(char ch)
{
   if(ch>='0' && ch<='9')
          return 1;
   else
           return 0;
}
```

OR

```
int isoperand(char ch)
{
   return (ch>='0' && ch<='9');
}
```

```
float calculate(float op1,float op2,char ch)
{
       switch(ch)
       {
          case '+':
                    return op1 + op2;

          case '-':
                    return op1 - op2;
          case '*':
                    return op1 * op2;

          case '/':
                    return op1 / op2;
          case '%':
                    return fmod(op1,op2);
          case '$':
                    return pow(op1,op2);
          default:
                     return 0.0;
       }
}
```