```
else if ((operator exp = '(')
        Push (exp);
    Elseif ib (exch = ')')
        while (exp ==
    { while (exp(x=pop()) != ))
        {
            pop()
        }
    Else if ( priority[stack[top]] >= priority(exp) )
        {
            pop();
        Push(exp)
        }
    eft
}
}
if (top = -1)
{
while (stack[top] != -1)
{
pop()
}
if (x=0)
print res
else
invalid
}
```

WEEK 2

1] WAP to convert a given valid parathesized infix arithmetic expression to postfix expression consists of single character operands and binary operations

+(PLUS), -(minus), * (multiply) and \ (divides)

write the pseudocode in your observation ~~and get it corrected from your batch faculty~~.

```
STACK SIZE = 20, top = -1
push (int x)
  {
    stack [++top] = x ;
  }
pop ()
  {
    If stack[0] top = -1 then return -1
      Else
        return top--;
  }
Priority()
    {
      If ('(') then return 0
      If ('+' or '-') then return 1
      If ('*' or '/') then return 2
    }
main()                    exp
{ for i in strlen
If (exp(i) = '(')
      k = 0
else if (exp(i) = ')')
      j = 0
}
while (exp != null)
  {
    if (exp is alnum(exp))
      {
        newexp[k++] = exp
      }
```

else if (exp[i] == '(')
    push(exp[i]);
else if (exp[i] == ')')
{
    while (exp[top--] != '(')
    {
        pop();
    }
}
else if (priority(stack[top]) >= priority(exp[i]))
{
    pop();
    push(exp[i]);
}
else
{
    top++;
}
}

while (top != -1)
{
    while (stack[top] != -1)
    {
        pop();
    }
}

if (i == j)
    print res
else
    invalid
}

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
char stack[20];
int top=-1;
int count1=0, count2=0;
void push(char x)
{
  stack[++top]=x;
}
char pop()
{
  if(top==-1)
    return -1;
  else
    return stack[top--];
}
int priority(char x)
{
  if(x=='(')
    return 0;
  if(x=='+'||x=='-')
    return 1;
  if(x=='*'||x=='/')
    return 2;
}
int main()
{
  int j=0;
  char exp[100];
```

```
char res[100];
char *e, x;

Printf ("Enter the expression : ");
scanf ("%s", exp);
Printf("\n");
e = exp;
int r = strlen(exp);
Printf("\n");
e = exp;
int r=strlen(exp);
for(int i=0; i<r; i++)
{
  if (exp[i] == '(')
      count1++;
  else if (exp[i] == ')')
      count2++;
}
while (*e != '\0')
{
  if (isalnum(*e))
    res[j++] = *e;
  else if (*e == '(')
    push (*e);
  else if (*e == ')')
  {
    while ((x = pop()) != '(')
      res[j++] = x;
  }
  else
  {
    while (priority(stack[top]) >= priority(*e))
```

```
            res[j++] = pop();
            push(*e);
        }
    e++;
}
while (top!=-1)
        res[j++] = pop();
if (count1 == count2)
    printf("%s", res);
else
  {
    printf("Invalid expression\n");
     exit(0);
  }
return 0;
}
```

OUTPUT

Enter the expression : (((8+3)/4)-1)

83+4/1-