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

int nodecount = 0;

struct Node
{
   struct Node *Next;
   int node_no;
   char nextA[3];
   char nextB[3];
};

struct Node *Head, *p, *Temp;

struct Node *create()
{
   int c;
   Temp = (struct Node *)malloc(sizeof(struct Node));
   Temp->node_no = nodecount;
nodecount++;
   Temp->Next = NULL;
   return Temp;
}

void update(struct Node *p, char Ch)
{
   char c;
   if (Ch == 'a')
   {
      c = p->node_no + '0'; //int to char conversion
strncat(p->nextA, &c, 1);
   }
   else if (Ch == 'b')
   {
      c = p->node_no + '0';
```

```
strncat(p->nextB, &c, 1);
   }
}

int main(void)
{
   char prev;
   char regex[100], Ch[5];
   int s, j, i;
   char c;

printf("Enter the regular expression");
scanf("%s", regex);

   Head = create();
   p = Head;
   for (i = 0; i<strlen(regex); i++)
   {
      if (regex[i] == '(')
      {
         s = 0;
Ch[0] = Ch[1] = Ch[2] = '\0';
         char prev;
         while (regex[i] != ')')
         {
            if (regex[i] == 'a' || regex[i] == 'b')
            {
               Ch[s] = regex[i];
               s++;
            }
i++;
         }
         if (regex[i + 1] == '*')
         {
            s = 0;
            while (Ch[s] != '\0')
            {
update(p, Ch[s]);
```

```
                s++;
            }
i++;
         }
         if (regex[i + 1] == '+')
         {
            Temp = create();
            c = Temp->node_no + '0';
strcat(p->nextA, &c);
strcat(p->nextB, &c);
            p->Next = Temp;
            s = 0;
            p = Temp;
            while (Ch[s] != '\0')
            {
update(p, Ch[s]);
                s++;
            }
         }
      }
      else if (regex[i] == '+')
      {
prev = regex[i - 1];
update(p, prev);
      }
      else if (regex[i + 1] == '*')
      {
prev = regex[i];
update(p, prev);
i++;
      }
      else
      {
         // a or b
         Temp = create();
         p->Next = Temp;
         c = Temp->node_no + '0';
         if (regex[i] == 'a')
```

```
strncat(p->nextA, &c, 1);
          else if (regex[i] == 'b')
strncat(p->nextB, &c, 1);
          p = Temp;
       }
    }

printf("\nTransition table");
printf("\n+----------------------------------------------------+");
printf("\n|\tState\t|\ta\t|\tb\t|");
printf("\n|----------------------------------------------------|");

    p = Head;
    for (j = 0; j <nodecount; j++)
    {
printf("\n|\t%d", p->node_no);
       if (p->nextA[0] == 0)
printf("\t|\tnull");
       else
       {
printf("\t|\t");
          for (i = 0; i< p->nextA[i] != '\0'; i++)
          {
printf("%c,", p->nextA[i]);
          }
       }
       if (p->nextB[0] == 0)
printf("\t|\tnull\t|");
       else
       {
printf("\t|\t");
          for (i = 0; p->nextB[i] != '\0'; i++)
printf("%c,", p->nextB[i]);
printf("\t|");
       }
       p = p->Next;
    }
printf("\n+----------------------------------------------------+");
```

```
    return 0;
}
```