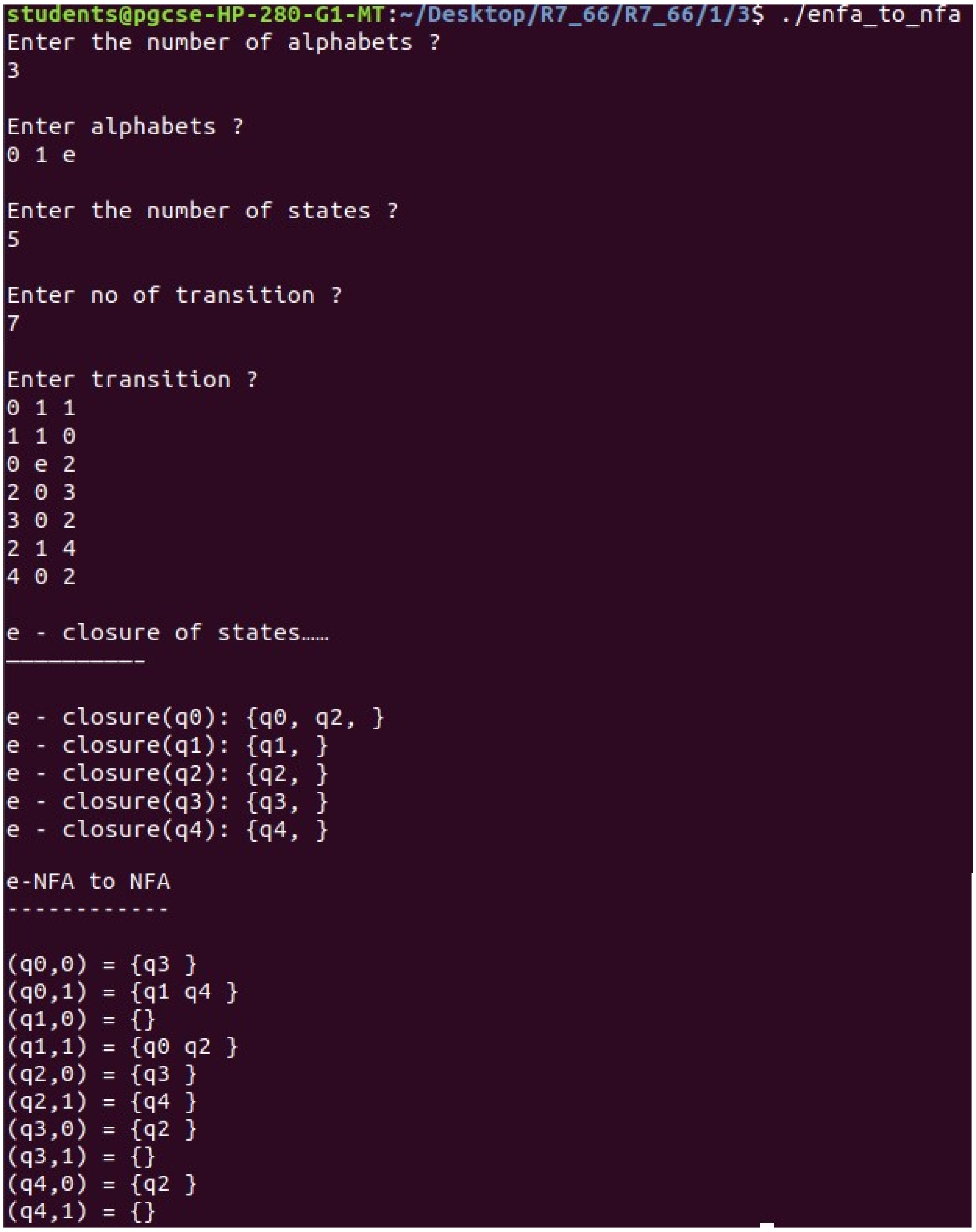
**Program Code**

```
#include<stdio.h>

#include<stdlib.h>

struct node {
  int st;
  struct node *link;
};

int state, alpha, s, no_transition,c,r, buffer[20];
char alphabet[20];

int e_closure[20][20] = { 0 };
struct node *transition[20][20] = { NULL };

void print_e_closure(int i)
{
  int j;
  printf("{");
    for (j = 0; e_closure[i][j] != -1; j++)
      printf("q%d, ",e_closure[i][j]);
     printf("}");
}


int findalpha(char y)
{
  for (int i = 0; i < alpha; i++)
  {
    if (alphabet[i] == y)
    {
      return i;
    }
  }
  return (999);
}

void add_closure(int k, int visited[],int t)
{
  for (int j = 0; e_closure[k][j] != -1; j++)
  {
      int flag=0;
      for(int i=0;i<t;i++)
      {
        if(e_closure[k][j]==visited[i])
        {
          flag=1;
          break;
        }
      }
      if(!flag)
      {
        visited[t++]=e_closure[k][j];
        printf("q%d ",visited[t-1]);
      }
    }
}
```

```
void enfa_to_nfa(int i,char y)
{
  int t=0;
  int visited[20];
  if(y!='e')
  {
    printf("\n(q%d,%c) = {",i,y);
    for (int j = 0; e_closure[i][j] != -1; j++)
    {
      int u=findalpha(y);
      if (u == 999)
      {
        printf("error\n");
        exit(0);
      }
      if(transition[e_closure[i][j]][u]!=NULL)
      {
        int k=transition[e_closure[i][j]][u]->st;
        add_closure(k,visited,t);
      }
    }
    printf("}");
  }
}

void findclosure(int x, int sta)
{
  struct node * temp;
  int i;
  if (buffer[x])
    return;
  e_closure[sta][c++] = x;
  buffer[x] = 1;
  if (alphabet[alpha - 1] == 'e' && transition[x][alpha - 1] !=
NULL)
  {
    temp = transition[x][alpha - 1];
    while (temp != NULL)
    {
      findclosure(temp->st, sta);
      temp = temp -> link;
    }
  }
}


void insert_trantbl(int r, char y, int s)
{
  int j;
  struct node * temp;
  j = findalpha(y);
  if (j == 999)
  {
    printf("error\n");
    exit(0);
  }
  temp = (struct node * ) malloc(sizeof(struct node));
  temp->st = s;
  temp->link = transition[r][j];
  transition[r][j] = temp;
```

```c
}



void main() {
  int i, j, k, m, t, n;
  char y;

  struct node *temp;
  printf("Enter the number of alphabets ? \n");
  scanf(" %d", &alpha);
  printf("\nEnter alphabets ? \n");
  for (i = 0; i < alpha; i++)
  {
    scanf(" %c",&alphabet[i]);
  }
  printf("\nEnter the number of states ? \n");
  scanf("%d", &state);
  printf("\nEnter no of transition ? \n");
  scanf("%d", &no_transition);
  printf("\nEnter transition ? \n");
  for (i = 0; i < no_transition; i++)
  {
    scanf("%d %c%d",&r,&y,&s);
    insert_trantbl(r,y,s);
  }
  printf("\n");
  printf("e - closure of states……\n");
  printf("—————————-\n");

  for (i = 0; i<state; i++)
  {
    c = 0;
    for (j = 0; j < 20; j++)
    {
      buffer[j] = 0;
      e_closure[i][j] = -1;
    }
    findclosure(i, i);
    printf("\ne - closure(q%d): ", i);
    print_e_closure(i);
  }

  printf("\n\ne-NFA to NFA");
  printf("\n------------\n");
  for(int i=0;i<state;i++)
  {
    for(int j=0;j<alpha;j++)
    {
      enfa_to_nfa(i,alphabet[j]);
    }
  }
  printf("\n");
}
```

**Output**


```
students@pgcse-HP-280-G1-MT:~/Desktop/R7_66/R7_66/1/3$ ./enfa_to_nfa
Enter the number of alphabets ?
3

Enter alphabets ?
0 1 e

Enter the number of states ?
5

Enter no of transition ?
7

Enter transition ?
0 1 1
1 1 0
0 e 2
2 0 3
3 0 2
2 1 4
4 0 2

e - closure of states……
__________

e - closure(q0): {q0, q2, }
e - closure(q1): {q1, }
e - closure(q2): {q2, }
e - closure(q3): {q3, }
e - closure(q4): {q4, }

e-NFA to NFA
------------

(q0,0) = {q3 }
(q0,1) = {q1 q4 }
(q1,0) = {}
(q1,1) = {q0 q2 }
(q2,0) = {q3 }
(q2,1) = {q4 }
(q3,0) = {q2 }
(q3,1) = {}
(q4,0) = {q2 }
(q4,1) = {}
```