Exp. No. 10

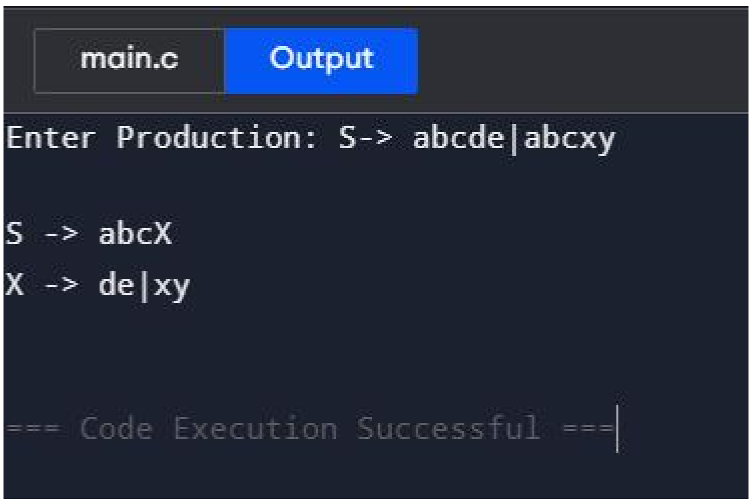
Implement a C program to eliminate left factoring from a given CFG.

S → iEtS / iEtSeS / a

E → b

PROGRAM:

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

int main() {
   char gram[50], part1[25], part2[25], modifiedGram[25],
newGram[25];
   int i, j = 0, k = 0, pos = 0;

   // Input production
   printf("Enter Production: S-> ");
   fgets(gram, sizeof(gram), stdin);

   // Remove newline character from fgets input
   gram[strcspn(gram, "\n")] = 0;

   // Extract part1 and part2
   for (i = 0; gram[i] != '|' && gram[i] != '\0'; i++, j++)
      part1[j] = gram[i];
   part1[j] = '\0';

   if (gram[i] == '|') i++;  // Move past '|'

   for (j = 0; gram[i] != '\0'; i++, j++)
      part2[j] = gram[i];
   part2[j] = '\0';

   // Find common prefix
   for (i = 0; part1[i] == part2[i] && part1[i] != '\0'; i++) {
      modifiedGram[k++] = part1[i];
      pos = i + 1;
   }

   // If no common prefix, no need for factoring
   if (k == 0) {
      printf("\nNo common prefix found. Left factoring is not
needed.\n");
      return 0;
   }

   // Create modified production
   modifiedGram[k] = 'X';
   modifiedGram[k + 1] = '\0';

   // Create new production
   j = 0;
   for (i = pos; part1[i] != '\0'; i++)
      newGram[j++] = part1[i];

   if (newGram[j - 1] != '|')  // Avoid duplicate '|'
      newGram[j++] = '|';

   for (i = pos; part2[i] != '\0'; i++)
      newGram[j++] = part2[i];

   newGram[j] = '\0';

   // Print the result
   printf("\nS -> %s", modifiedGram);
   printf("\nX -> %s\n", newGram);

   return 0;
}
```



```
Enter Production: S-> abcde|abcxy

S -> abcX
X -> de|xy


=== Code Execution Successful ===
```