Name : Salman Ahmed Khan
Roll No: CSE-20-03
Compiler Design Lab
Date :26-04-2024

Q1: Program to identify white spaces in a sentence ?

```
#include <stdio.h>

int main()
{
   char ch;
   int count = 0;

   printf("Enter a sentence: ");

   while ((ch = getchar()) != '\n')
   {
      if (ch == ' ')
      {
         count++;
      }
   }

   printf("Number of white spaces: %d\n", count);

   return 0;
}
```

Q2: Program to count the number of words(takens) in a string ?

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

int main()
{
   char ch, lastChar = '\0';
   int count = 0;
   bool firstWordFound = false, lastCharIsSpace = false;

   printf("Enter a sentence: ");

   while ((ch = getchar()) != '\n') // Read the characters until a new line is found
   {
      if (ch == ' ') // If the character is a space
      {
         if (lastChar != ' ' && firstWordFound) // last character was space and next
character is symbol then new word is found, so increment the count
         {
            count++;
            lastCharIsSpace = true;
         }
      }
      else // If the character is not a space
      {
         firstWordFound = true;
         lastCharIsSpace = false;
      }

      lastChar = ch;
   }

   if (firstWordFound && !lastCharIsSpace) // Count the last word if it's not
followed by a space
   {
      count++;
   }

   printf("Number of tokens: %d\n", count);

   return 0;}
```

Q3: Token Length ?

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

```
#include <stdbool.h>

int main()
{
   char ch;
   int count = 0, tokenLength = 0;
   bool isSpace = true;

   printf("Enter a sentence: ");

   while ((ch = getchar()) != '\n') // Read the characters until a new line is found
   {
      if (ch == ' ') // If the character is a space
      {
         if (!isSpace) // End of a token
         {
            count++; // Increment the word count
            printf("Length of token %d: %d\n", count, tokenLength);
            tokenLength = 0; // Reset the length for the next token
         }
         isSpace = true;
      }
      else // If the character is not a space
      {
         tokenLength++; // Increment the length of the current token
         isSpace = false;
      }
   }

   if (!isSpace) // If the last character is not a space, count the last word
   {
      count++;
      printf("Length of token %d: %d\n", count, tokenLength);
   }

   printf("Number of tokens: %d\n", count);

   return 0;
}
```

Q4: Program to check if a number is a palindrome ?

```
// 30th November 2023

#include <stdio.h>

int main()
{
   int n, r, rev = 0, m;

   printf("Enter a number: ");
   scanf("%d", &n);

   m = n; // Preserving the value of n before it becomes 0. We need this value to
compare with the reversed number

   while (n != 0)
   {
      r = n % 10;
      n = n / 10;
      rev = rev * 10 + r;
   }

   printf("Reverse of %d is %d\n", m, rev);

   if (rev == m)
      printf("The number is palindrome.\n");
   else
      printf("The number is not palindrome.\n");

   return 0;
}
```

Q5: Program to check if a string is a palindrome - using loops ?

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

```
#include <string.h>

int main()
{
   char str[100];
   printf("Enter a string you want to check: ");
   gets(str); // Reads the line with spaces

   int len = strlen(str);
   int i = 0; // iterating from the start
   int j = len - 1; // iterating from the end
   int flag = 0;

   while (i < j)
   {
      if (str[i] != str[j])
      {
         flag = 1;
         break;
      }
      i++;
      j--;
   }

   if (flag == 0)
   {
      printf("The string is palindrome");
   }
   else
   {
      printf("The string is not palindrome");
   }
}
```

Q6: Write a program that checks if a sentence is a palindrome or not ?

// This program ignores the white spaces in the sentence. For example, "ab b a" is a palindrome although it is not a palindrome if we consider the white spaces.

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

// Function to remove all spaces from a given string
void removeSpaces(char *str) // *str is a pointer to the first character of the string
{
   int i = 0, j = 0;
   while (str[i])
   {
      if (str[i] != ' ')
      {
         str[j++] = str[i];
      }
      i++;
   }
   str[j] = '\0'; // Null-terminate the string
}

int main()
{
   char str[100];
   printf("Enter a string you want to check: ");
   gets(str); // Reads the line with spaces

   removeSpaces(str); // Remove spaces from the string

   int len = strlen(str);
   int i = 0;       // iterating from the start
   int j = len - 1; // iterating from the end
   int flag = 0;

   while (i < j)
   {
      if (str[i] != str[j])
      {
         flag = 1;
         break;
      }
      i++;
      j--;
   }

   if (flag == 0)
```

```
    {
        printf("The string is a palindrome\n");
    }
    else
    {
        printf("The string is not a palindrome\n");
    }
}
```

Q7: Create a program which reads a text file and identify the identifiers in a given string ?

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

```
int main()
{
   FILE *fp;
   char myString[100];
   char str[100];
   int i = 0;
   printf("Enter a string:");
   gets(str);

   while (i < 100)
   {
      if (str[i] == ' ')
         str[i] = '\n';
      i++;
   }
   printf("%s", str);

   printf("\nIdentifiers in the file are:\n");

   fp = fopen("Rules.txt", "r");
   while (fgets(myString, 100, fp))
   {
      printf("%s", myString);
   }

   // if
}
```

Q8: Write a program in C for the given grammar ?

```
// ( or {   ->  4
// ) or }   ->  5
// Digit    ->  6
// +        ->  2
// *       ->  3

#include <stdio.h>
#include <string.h>
#include <ctype.h> // For isdigit()

void grammarRules(const char *input, char *output) // const prevents the function
from modifying the input string
{
   for (int i = 0; input[i] != '\0'; i++)
   {
      if (input[i] == '(' || input[i] == '{')
      {
         output[i] = '4';
      }
      else if (input[i] == ')' || input[i] == '}')
      {
         output[i] = '5';
      }
      else if (isdigit(input[i]))
      {
         output[i] = '6';
      }
      else if (input[i] == '+')
      {
         output[i] = '2';
      }
      else if (input[i] == '*')
      {
         output[i] = '3';
      }
      else
      {
         output[i] = input[i];
      }
   }
   output[strlen(input)] = '\0';
}

int main()
{
   char inputString[100];
   char outputString[100];
   printf("Enter a string: ");

fgets(inputString, sizeof(inputString), stdin); // Read a string from the user, fgets
reads until newline or EOF and stores the newline character in the string, stdin is
the standard input stream

 int index_of_newline = strcspn(inputString, "\n"); // return the index when the
newline character is found
inputString[index_of_newline] = 0;  // Replace the newline character with null
character (0 or '\0' are the same thing in C)

    grammarRules(inputString, outputString);
    printf("Converted string: %s\n", outputString);

    return 0;
}
```

Q9: Check whether a string is a substring of a given parent string using Pointers ?

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

```
int main()
{
   char parent[100], substring[100];
   char *p, *s;
   int i, j;
   int found=0;

   printf("Enter the parent string: ");
   gets(parent);
   printf("Enter the substring: ");
   gets(substring);

   p = parent;
   s = substring;

   for (i = 0; i < strlen(parent); i++)
   {
      j = 0; // to iterate through the substring
      if (*p == *s)
      {
         while (*(s + j) != '\0')
         {
            if (*(p + j) == *(s + j))
            {
               j++;
            }
            else
            {
               break;
            }
         }
         if (*(s + j) == '\0')
         {
            found = 1;
            break;
         }
      }
      p++;
   }

   if (found == 1)
   {
      printf("The substring is present in the parent string\n");
```

```
    }
    else
    {
        printf("The substring is not present in the parent string\n");
    }

    return 0;
}
```

Q10: Write a program in C for the given grammar ?

```
// ( or {   ->  4
// ) or }   ->  5
// Digit    ->  6
// +        ->  2
// *        ->  3
// Further in the output if you encounter:
// consecutive 6s -> 6
// 626 ->  6
// 636 ->  6
// 465 ->  6

// Example Question:
// (D+D) || (D*D) || (D)
// {2 + (3 + 4) +5}
// {2 + 7 + 5}
// {14} - Reseamble (D)

#include <stdio.h>
#include <string.h>
#include <ctype.h> // For isdigit()

void grammarRules(const char *input, char *output) // const prevents the function
from modifying the input string
{
    for (int i = 0; input[i] != '\0'; i++)
    {
        if (input[i] == '(' || input[i] == '{')
        {
            output[i] = '4';
        }
        else if (input[i] == ')' || input[i] == '}')
        {
            output[i] = '5';
        }
        else if (isdigit(input[i]))
        {
            output[i] = '6';
        }
        else if (input[i] == '+')
        {
            output[i] = '2';
        }
        else if (input[i] == '*')
        {
            output[i] = '3';
        }
        else
        {
            output[i] = input[i];
        }
    }
    output[strlen(input)] = '\0';
}

void reduceRule(const char *input, char *output)
{
    int j = 0; // Index for the output string

    for (int i = 0; input[i] != '\0';)
    {
        // Check for '465' pattern
        if (input[i] == '4' && input[i + 1] == '6' && input[i + 2] == '5')
        {
            output[j++] = '6';
            i = i + 3; // Skip over processed characters
        }

        // Check for '626' and '636' patterns
        else if (input[i] == '6' && (input[i + 1] == '2' || input[i + 1] == '3') && input[i
+ 2] == '6')
        {
            output[j++] = '6';
            i = i + 3; // Skip over processed characters
        }

        // Check for consecutive '6's
        else if (input[i] == '6' && input[i + 1] == '6')
        {
            output[j++] = '6';
            i = i + 2; // Skip over processed characters
        }

        // Default case, just copy the characte
 else
        {
            output[j++] = input[i++];
        }
    }
    output[j] = '\0';
}

int main()
```

```
{
   char inputString[100];
   char tokenString[100];
   char resultString[100];
   char temp[100];

   printf("Enter a string: ");
   fgets(inputString, sizeof(inputString), stdin); // Read a string from the user, fgets
reads until newline or EOF and stores the newline character in the string, stdin is
the standard input stream

   int index_of_newline = strcspn(inputString, "\n"); // return the index when the
newline character is found
   inputString[index_of_newline] = 0;              // Replace the newline character
with null character (0 or '\0' are the same thing in C)

   grammarRules(inputString, tokenString);
   printf("Converted string: %s\n", tokenString);

   // reduceRule(tokenString, resultString); // delete this line after explaining

   strcpy(temp, tokenString); // Initialize temp with the Token string
   while (1)
   {
      reduceRule(temp, resultString);

      // Check if further simplification is possible
      if (strcmp(temp, resultString) == 0)
      {
         break; // Stop if no changes
      }

      // Otherwise, update temp and print the intermediate result
      strcpy(temp, resultString);
      printf("Middle Iterations: %s\n", resultString);
   };

   printf("Checked string: %s\n", resultString);

   return 0;
}
```

Q11: Write a program in C to check whether a statement entered is a comment or not ?

```
// // This is a comment
// /* This is a multi-line comment */


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

int main()
{
   char statement[100];
   int length = 0;

   printf("Enter a statement: ");
   gets(statement);

   length = strlen(statement);

   // Checch if a statement starts with //
   if (statement[0] == '/' && statement[1] == '/')
   {
      printf("The statement is a single line comment.\n");
   }

   // Check if a statement starts with /* and ends with */
   else if ((statement[0] == '/' && statement[1] == '') && (statement[length - 2] ==
'' && statement[length - 1] == '/'))
   {
      printf("The statement is a multi-line comment.\n");
   }

   // Default case
   else
   {
      printf("The statement is not a comment.\n");
   }
   return 0;
}
```