Compiler Design: Lab 1

## Anirudh Sathish, CS20B1125

1. 1. Write a simple C code to remove spaces and new line enter characters of a given input C program.  
     
   Input: A text file with a C program  
     
   Output: A text file of the above C program without any space between words and the whole program is in a single line.

**Code:**

/\*

       Compiler Design

       @Author : Anirudh Sathish

       @Roll No : CS20B1125

       1. Write a simple C code to remove spaces and new line enter

        characters of a given input C program.

        Input: A text file with a C program

        Output: A text file of the above C program without any space

        between words and the whole program is in a single line.

\*/

// Defining the header files

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

/\*Defining a function to remove spaces and new line character

And return the character new array \*/

char \*removeSpace(char \*msg , long int length)

{

    /\* Intializing the new character array , which will hold the resultant value\*/

    char \*newMsg;

    if(length>0)

    {

        newMsg = malloc(length\*sizeof(char));

    }

    int i = 0 ;

    int j = 0 ;

    //Parsing through the whole text

    for(; i < length; i++)

    {

        if(msg[i] != ' ')

        {

            if(msg[i] != 10)

            {

                // Adding the character to the resultant

                // If it is does not have space or newline

                newMsg[j] = msg[i];

                j++;

            }

        }

    }

    //Returning the required result

    return newMsg;

}

int main()

{

    // Preparing to take text input

    FILE \*ptr;

    char \*inputString;

    long int num;

    // ptr is is a FILE pointer , it points to the text of

    // the desired file

    // Replace containscode.txt by any textfile neccesary

    // Currently ensure , to keep it in the same folder

    // as the source code

    // This can be overcome by mentioning path

    ptr = fopen("containscode.txt", "r");

    if(ptr == NULL)

        return 1;

    // To find the total length of the text file

    fseek(ptr, 0L, SEEK\_END);

    num = ftell(ptr);

    fseek(ptr, 0L, SEEK\_SET);

    // Initalising a string to store the text

    inputString = (char\*)calloc(num, sizeof(char));

    if(inputString == NULL)

        return 1;

    //Reading into string , and further closing the file ptr

    fread(inputString, sizeof(char), num, ptr);

    fclose(ptr);

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    printf("Text as in file \n");

    printf(inputString);

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    printf("Output resultant text would be \n \n ");

    // The string to hold the reultant

    char \*outputString;

    outputString = removeSpace(inputString,num);

    printf(outputString);

    printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    // Printing out the resultant to a textfile

    FILE \*myPtr;

    myPtr = fopen("output\_q1.txt","w");

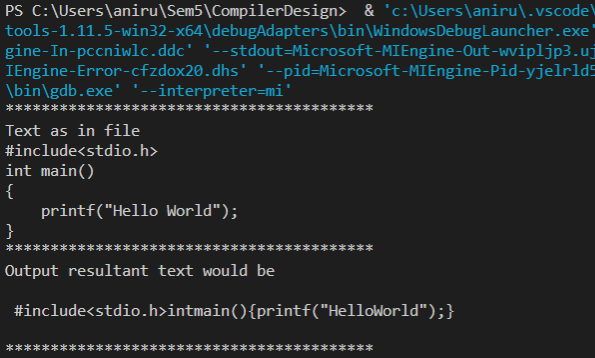
    fprintf(myPtr,outputString);

    fclose(myPtr);

    return 0;

}

**Output:**

****

1. Write a simple C code to remove spaces and new line enter characters and **comment lines** of a given input C program.  
     
   Input: A text file with a C program  
     
   Output: A text file of the above C program without any space between words and the whole program is in a single line. **All comments to be removed from the input.**

**Code:**

/\*

       Compiler Design

       @Author : Anirudh Sathish

       @Roll No : CS20B1125

       2. Write a simple C code to remove spaces and new line

       enter characters and comment lines of a given input C program

       Input: A text file with a C program

       Output: A text file of the above C program without any space

       between words and the whole program is in a single line.

       All comments to be removed from the input.

\*/

// Defining the header files

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

/\* Function to deal with comments , both single and multi line comm

ents \*/

int foundbackslash(char \*msg , int i , long length)

{

    // Intialiing variables

    int index = i;

    int front = index+1;

    int j ;

    index++;

    front++;

    int change;

    // Dealing with the case of single line comments

    if(msg[index] == '/')

    {

        //Comments come to end when line ends

        // So keeping track of end of line

        while(msg[index]!= 10)

        {

            index++;

        }

        return index;

    }

    // Dealing with the case of multiline comments

    else if(msg[index] =='\*')

    {

        index++;

        front++;

        change = 0 ;

        //Dealing with end of multiline comments

        while(msg[index]!='\*' || msg[front] != '/')

        {

            if(msg[front] == '/')

            {

                printf("End");

                return index;

            }

            index++;

            front++;

            change = 1 ;

        }

        // Useful To check if end of line was found or not

        if(change == 1)

        {

            if(msg[index] == '\*' && msg[front] =='/')

            {

                index = index+2;

                return index;

            }

        }

        return index;

    }

    return i;

}

// Function to deal with spaces , end lines and comments

char \*removeSpace(char \*msg , long int length)

{

    // Initalising array which has to hold the resultant

    char \*newMsg;

    if(length>0)

    {

        newMsg = malloc(length\*sizeof(char));

    }

    int i = 0 ;

    int j = 0 ;

    // Checking for end of line and blank space

    // While parsing

    for(; i < length; i++)

    {

        if(msg[i] != ' ')

        {

            if(msg[i] == '/')

            {

                // Noticed a black slash

                i = foundbackslash(msg, i,length);

            }

            if(msg[i] != 10)

            {

                newMsg[j] = msg[i];

                j++;

            }

        }

    }

    return newMsg;

}

int main()

{

     // Preparing to take text input

    FILE \*ptr;

    char \*inputString;

    long int num;

    // ptr is is a FILE pointer , it points to the text of

    // the desired file

    // Replace containscodewithcoments.txt by any textfile neccesary

    // Currently ensure , to keep it in the same folder

    // as the source code

    // This can be overcome by mentioning path

    ptr = fopen("containscodewithcomments.txt", "r");

    if(ptr == NULL)

        return 1;

    // To find the total length of the text file

    fseek(ptr, 0L, SEEK\_END);

    num = ftell(ptr);

    fseek(ptr, 0L, SEEK\_SET);

    // Initalising a string to store the text

    inputString = (char\*)calloc(num, sizeof(char));

    if(inputString == NULL)

        return 1;

    //Reading into string , and further closing the file ptr

    fread(inputString, sizeof(char), num, ptr);

    fclose(ptr);

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    printf("Text as in file \n");

    printf(inputString);

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    printf("Output resultant text would be \n \n ");

    char \*outputString;

    outputString= removeSpace(inputString,num);

    printf(outputString);

    printf("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    // To store the processed text in a text file

    FILE \*myPtr;

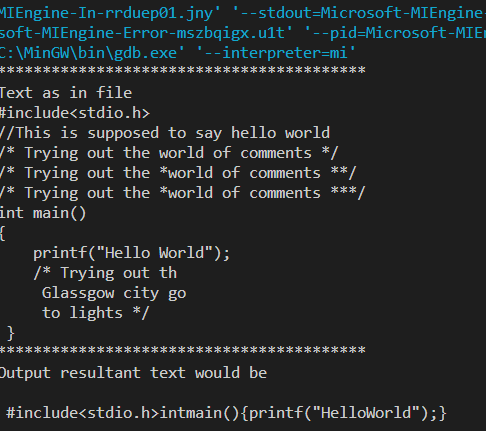
    myPtr = fopen("output\_q2.txt","w");

    fprintf(myPtr,outputString);

    fclose(myPtr);

    return 0;

}



1. Given a C program text file as input, write a C program and separate all Words and put all of them in an array.  
     
   For example if the text file has the following line "My name = Ramesh+Suresh" then the output will be  
     
   a[0]=My, a[1]=name, a[2]==, a[3]=Ramesh+Suresh  
     
   You can print the array as output on the screen

**Code:**

/\*

       Compiler Design

       @Author : Anirudh Sathish

       @Roll No : CS20B1125

       3. Given a C program text file as input, write a C program

       and separate all Words and put all of them in an ar

       ray.

       For example if the text file has the following line

       "My name = Ramesh+Suresh" then the output will be

        a[0]=My, a[1]=name, a[2]==, a[3]=Ramesh+Suresh

        You can print the array as output on the screen.

\*/

// Defining the header files

#include <stdio.h>

#include <stdlib.h>

#include<string.h>

char \*storeToken(char \*msg ,int count , int highest\_len, int length)

{

    int msg\_len;

    // 2D Character array , that consists of all the tokens(words)

    char \*resultant;

    //Setting the size of the array

    int totalSize = highest\_len\*count;

    if(totalSize > 0)

    {

        resultant = malloc(totalSize\*(sizeof(char)));

    }

    // Storing the value as a resultant array

    int t = 0 , i , j;

    if(t <length)

    {

        for (i = 0; i < count; i++)

        {

            msg\_len = 0;

            for ( j = 0; j < highest\_len;)

            {

                if(msg[t] != ' ' && msg[t] != 10)

                {

                    // Char is stored in resultant

                    // If there is no space and

                    // end line

                    resultant[i\*highest\_len+j] = msg[t];

                    t++;

                    j++;

                    msg\_len++;

                }

                else

                {

                    t++;

                    break;

                }

            }

            // If the message length is 0 , dont count it

            // To counter act the increment

            // Perform a decrement

            if(msg\_len <= 0)

            {

                i = i-1;

            }

        }

    }

    return resultant;

}

// To print the tokens from the stored array

void printArrayToken(char \*resultant ,int count , int highest\_len)

{

    int i , j;

    // Printing out the resultant

    printf("The resultant is \n");

    for (i = 0; i < count; i++)

    {

        printf("Stored\_Array[%d] : ",i );

        for ( j = 0; j < highest\_len;j++)

        {

            printf("%c",resultant[i\*highest\_len+j]);

        }

        printf("\n");

    }

}

// Function to tokenize the textfile

char \*tokenize(char \*msg , long int length)

{

    char \*newMsg;

    /\* Allocating Space to the character array \*/

    if(length>0)

    {

        newMsg = malloc(length\*sizeof(char));

    }

    int i = 0 ,j = 0 ,highest\_len = 0, msg\_len ,count = 0 ;

    // Parsing the message

    for(; i < length; i++)

    {

        msg\_len = 0 ;

        // While there is no space or newLine

        while(msg[i] != ' ' && i < length && msg[i] != 10)

        {

            msg\_len++;

            i++;

        }

        // To find the highest length of the message

        if(msg\_len > highest\_len)

        {

            highest\_len = msg\_len;

        }

        // To keep track of the number of words

        if(msg\_len > 0)

        {

            count++;

        }

    }

    // To store the results in an array

    char \*resultant ;

    resultant = storeToken(msg,count,highest\_len,length);

    // Printing out the resultant

    printArrayToken(resultant,count,highest\_len);

    return resultant;

}

int main()

{

     // Preparing to take text input

    FILE \*ptr;

    char \*inputString;

    long int num;

    // ptr is is a FILE pointer , it points to the text of

    // the desired file

    // Replace containscode.txt by any textfile neccesary

    // Currently ensure , to keep it in the same folder

    // as the source code

    // This can be overcome by mentioning path

    ptr = fopen("containscode.txt", "r");

    if(ptr == NULL)

        return 1;

    // To find the total length of the text file

    fseek(ptr, 0L, SEEK\_END);

    num = ftell(ptr);

    fseek(ptr, 0L, SEEK\_SET);

    // Initalising a string to store the text

    inputString = (char\*)calloc(num, sizeof(char));

    if(inputString == NULL)

        return 1;

    //Reading into string , and further closing the file ptr

    fread(inputString, sizeof(char), num, ptr);

    fclose(ptr);

    printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    printf("Text as in file \n");

    printf(inputString);

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    printf("Output would be \n \n ");

    char \*tokenized\_text;

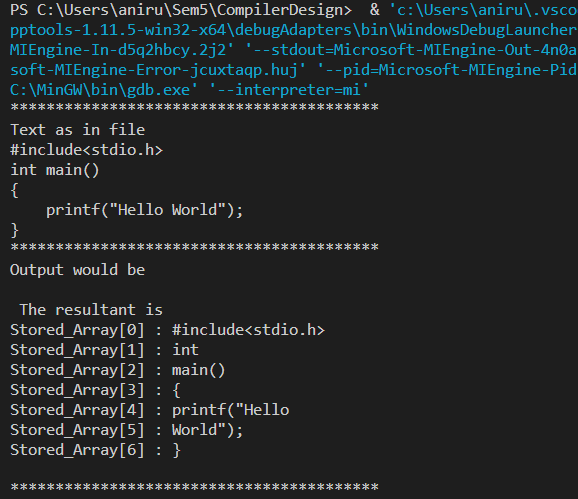
    tokenized\_text = tokenize(inputString,num);

    printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

    return 0;

}

**Output:**

****

1. Write a C program to convert a given simple mathematical expression into prefix notation. You can assume some simple conditions for input like only 3 to 4 components are there in the expression.  
     
   Example = If the input string is E=(X\*Y)-(4Z+X) then the output must be E=  - \* X Y + Z X.

**Code:**

/\*

       Compiler Design

       @Author : Anirudh Sathish

       @Roll No : CS20B1125

        4. Write a C program to convert a given simple mathematical

        expression into prefix notation. You can assume some simple

        conditions for input like only 3 to 4 components are there

        in the expression.

        Example = If the input string is

        E=(X\*Y)-(Z+X)

        then the output must be E=  - \* X Y + Z X.

\*/

// Defining the header files

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

// struct for linked list

struct list

{

    int data;

    struct list \*next;

}

\*top = NULL, \*node;

// For pushing value into stack

void push(int p)

{

    node = (struct list \*)malloc(sizeof(struct list));

    node->data = p;

    if (top == NULL)

    {

        node->next = NULL;

        top = node;

    }

    else

    {

    node->next = top;

    top = node;

    }

}

// To pop from a stack

void pop()

{

    if (top == NULL)

    {

        printf("Underflow ");

    }

    else

    {

        top = top->next;

        free(node);

        node = top;

    }

}

// Defining a priority order for the operators

int priority (char ch)

{

    if (ch == '^')

    {

        return 3;

    }

    if (ch == '\*' || ch == '/')

    {

        return 2;

    }

    if (ch == '+' || ch == '-')

    {

        return 1;

    }

    if (ch == ')')

    {

        return 0;

    }

}

// Function to find the prefix expression

void prefix(char \*a, char \*b)

{

    int j = 0;

    int i ;

    int term1 = -1;

    int term2;

    // Parsing through the expression

    for ( i = 0; a[i] != '\0'; i++)

    {

        if (a[i] == '+' || a[i] == '-' || a[i] == '\*' || a[i] == '/' || a[i] == '^')

        {

            term2 = priority(a[i]);

            if (top != NULL)

                term1 = priority(top->data);

            if (term1 < term2)

                push(a[i]);

            else if (term1 == term2)

            {

                if (term1 == 3)

                {

                    b[j] = a[i];

                    j++;

                }

            else

                push(a[i]);

            }

            else

            {

                while (term2 < term1 && top != NULL)

                {

                    b[j] = top->data;

                    pop();

                    j++;

                    if (top != NULL)

                        term1 = priority(top->data);

                }

                push(a[i]);

            }

        }

        else if (a[i] == ')')

            push(a[i]);

        else if (a[i] == '(')

        {

            while (top->data != ')')

            {

                b[j] = top->data;

                j++;

                pop();

            }

            pop();

        }

        else

        {

            b[j] = a[i];

            j++;

        }

    }

    b[j] = '\0';

}

// Function to reverse the given expression

void reverse(char \*str, char \*temp)

{

    int j = 1;

    int i ;

    temp[0] = ')';

    for (i = strlen(str) - 1; i >= 0; i--)

    {

        temp[j] = str[i];

        j++;

    }

    temp[j] = '(';

    temp[j + 1] = '\0';

}

int main()

{

    char string[100], temp [100];

    printf("Enter the String:- ");

    scanf("%s", string);

    // Reversing the expression

    reverse(string, temp);

    // Finding the prefix equivalent

    prefix(temp, string);

    printf("The prefix notation of the string is \n");

    for (int i = strlen(string) - 1; i >= 0; i--)

    {

        printf("%c", string[i]);

    }

    return 0;

}

**Output:**

