



IEEE
COMPUTER
SOCIETY



CERTIFICATE

OF PARTICIPATION

This certificate is awarded to

Siddhika shukla

For participating in the event “ROADMAP TO WEB DEVELOPMENT” by
IEEE GTBIT SB CS Chapter held on 10TH January 2022.

MR. MUKESH SAHU

BRANCH COUNSELLOR, IEEE GTBIT





IEEE
COMPUTER
SOCIETY



CERTIFICATE

OF PARTICIPATION

This certificate is awarded to

Siddhika shukla

For participating in the event “BASICS OF PYTHON” by IEEE GTBIT SB CS Chapter held on 26TH December 2021.

MR. MUKESH SAHU

BRANCH COUNSELLOR, IEEE GTBIT



```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
P  BT  WT  TAT
1  10  0   10
2  5   10  15
3  8   15  23
Average waiting time = 8.333333
Average turn around time = 16.000000
PS C:\college programs>
```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Enter number of process: 5
Enter Burst Time:
P1: 01
P2: 02
P3: 03
P4: 04
P5: 05
P      BT      WT      TAT
P1     1       0       1
P2     2       1       3
P3     3       3       6
P4     4       6      10
P5     5      10      15
Average Waiting Time= 4.000000
Average Turnaround Time= 7.000000
PS C:\college programs> █
```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Enter Number of Processes: 5
Enter Burst Time and Priority Value for Process 1: 1 3
Enter Burst Time and Priority Value for Process 2: 2 2
Enter Burst Time and Priority Value for Process 3: 3 1
Enter Burst Time and Priority Value for Process 4: 4 5
Enter Burst Time and Priority Value for Process 5: 5 4
Order of process Execution is
P4 is executed from 0 to 4
P5 is executed from 4 to 9
P1 is executed from 9 to 10
P2 is executed from 10 to 12
P3 is executed from 12 to 15

Process Id    Burst Time   Wait Time   TurnAround Time
P4            4             0           4
P5            5             4           9
P1            1             9          10
P2            2            10          12
P3            3            12          15
PS C:\college programs> █
```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Total number of process in the system: 2

Enter the Arrival and Burst time of the Process[1]
Arrival time is: 1

Burst time is: 8

Enter the Arrival and Burst time of the Process[2]
Arrival time is: 2

Burst time is: 4
Enter the Time Quantum for the process: 3

Process No          Burst Time          TAT          Waiting Time
Process No[2]        4                  8           4
Process No[1]        8                  11          3
Average Turn Around Time: 9.500000
Average Waiting Time: 3.500000

```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Enter no of pages:10
Enter the reference string:7 8 9 3 2 4 7 4 6 1
Enter no of frames:3

7
7     8
7     8     9
3     8     9
3     2     9
3     2     4
7     2     4
7     6     4
1     6     4

The no of page faults is 9
PS C:\college programs>

```

Incoming	t	Frame 1	t	Frame 2	t	Frame 3	
4		4		-		-	
1		4		1		-	
2		4		1		2	
4		4		1		2	
5		5		1		2	
Total Page Faults: 4							

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Enter number of frames: 3
Enter number of pages: 10
Enter page reference string: 7 8 9 3 2 1 6 5 4 9

7      -1      -1
7      8      -1
7      8      9
3      8      9
2      8      9
1      8      9
6      8      9
5      8      9
4      8      9
4      8      9

Total Page Faults = 9
PS C:\college programs>

```

```
Memory Management Scheme - Worst Fit
Enter the number of blocks:3
Enter the number of files:3

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4
File 3:5

File_no:      File_size :      Block_no:      Block_size:      Fragement
1            1              3                7                  6
2            4              1                5                  1
3            5              0                8                  0
```

```
Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File_no:      File_size :      Block_no:      Block_size:      Fragement
1            1              1                5                  4
2            4              3                7                  3
```

```
PS C:\college programs> █

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:7
Block 3:2
Enter the size of the files :-
File 1:4
File 2:3

File No File Size      Block No      Block Size      Fragment
1          4             1              5                  1
2          3             2              7                  4
```

```
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving

...Program finished with exit code 0
Press ENTER to exit console. █
```

```
Enter the number of Producers:2
Enter the number of Consumers:2
Enter buffer capacity:2
Successfully created producer 1
Producer 1 produced 29
Successfully created producer 2
Producer 2 produced 11
Successfully created consumer 1
Buffer:29 11
Consumer 3 consumed 11
Current buffer len: 1
Buffer:29
Consumer 2 consumed 29
Current buffer len: 0
Successfully created consumer 2
Producer 1 produced 3
Buffer:3
Consumer 2 consumed 3
Current buffer len: 0
Producer 2 produced 31
Producer 1 produced 27
Producer 1 produced 7
Buffer:31 27
```

```
Enter the number of process and resources
5 3
enter allocation of resource of all process 5x3 matrix
1 2 3
4 5 6
0 6 7
0 8 9
9 5 4
enter the max resource process required 5x3 matrix
0 1 0
1 2 3
2 3 4
3 4 5
4 5 6
enter the available resource 3 4 2

need resources matrix are
-1      -1      -3
-3      -3      -3
2       -3      -3
3       -4      -4
-5      0       2

available resource after completion
17      30      31
safe sequence are
p0      p1      p2      p3      p4
```

```
Enter the directory name:siddhika
Enter the number of files:3
Enter file name to be created:sid
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:shukla
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:ss
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:sk
Do you want to enter another file(yes - 1 or no - 0):0
Directory name is:siddhika
Files names are:
sid
shukla
ss
sk
PS C:\college programs> █
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 1
```

```
Enter name of directory -- siddhika
Directory created
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 2
```

```
Enter name of the directory -- siddhika
Enter name of the file -- sid
File created
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 5
```

```
Directory      Files
siddhika      sid
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 6
```

```
PS C:\college programs> █
```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc hierarchical.c -o hierarchical } ; if (?) { .\hierarchical }
Enter number of users: 1
Enter name: siddhika
Enter dir(1) or file(0): sid
```

```
Hierarchical
█
```

GURU TEGH BAHADUR INSTITUTE OF
TECHNOLOGY

COMPILER DESIGN LAB FILE

NAME: SIDDHIKA SHUKLA

ENROLLMENT NO.: 05113203121

BRANCH: IT-1, 5TH SEM

TITLE TABLE

PRACTICAL – 1

Introduction to Compiler Design.

The compiler is software that converts a program written in a high-level language (Source Language) to a low-level language (Object/Target/Machine Language/0, 1's).

A translator or language processor is a program that translates an input program written in a programming language into an equivalent program in another language. The compiler is a type of translator, which takes a program written in a high-level programming language as input and translates it into an equivalent program in low-level languages such as machine language or assembly language.

The program written in a high-level language is known as a source program, and the program converted into a low-level language is known as an object (or target) program. Without compilation, no program written in a high-level language can be executed. For every programming language, we have a different compiler; however, the basic tasks performed by every compiler are the same. The process of translating the source code into machine code involves several stages, including lexical analysis, syntax analysis, semantic analysis, code generation, and optimization.

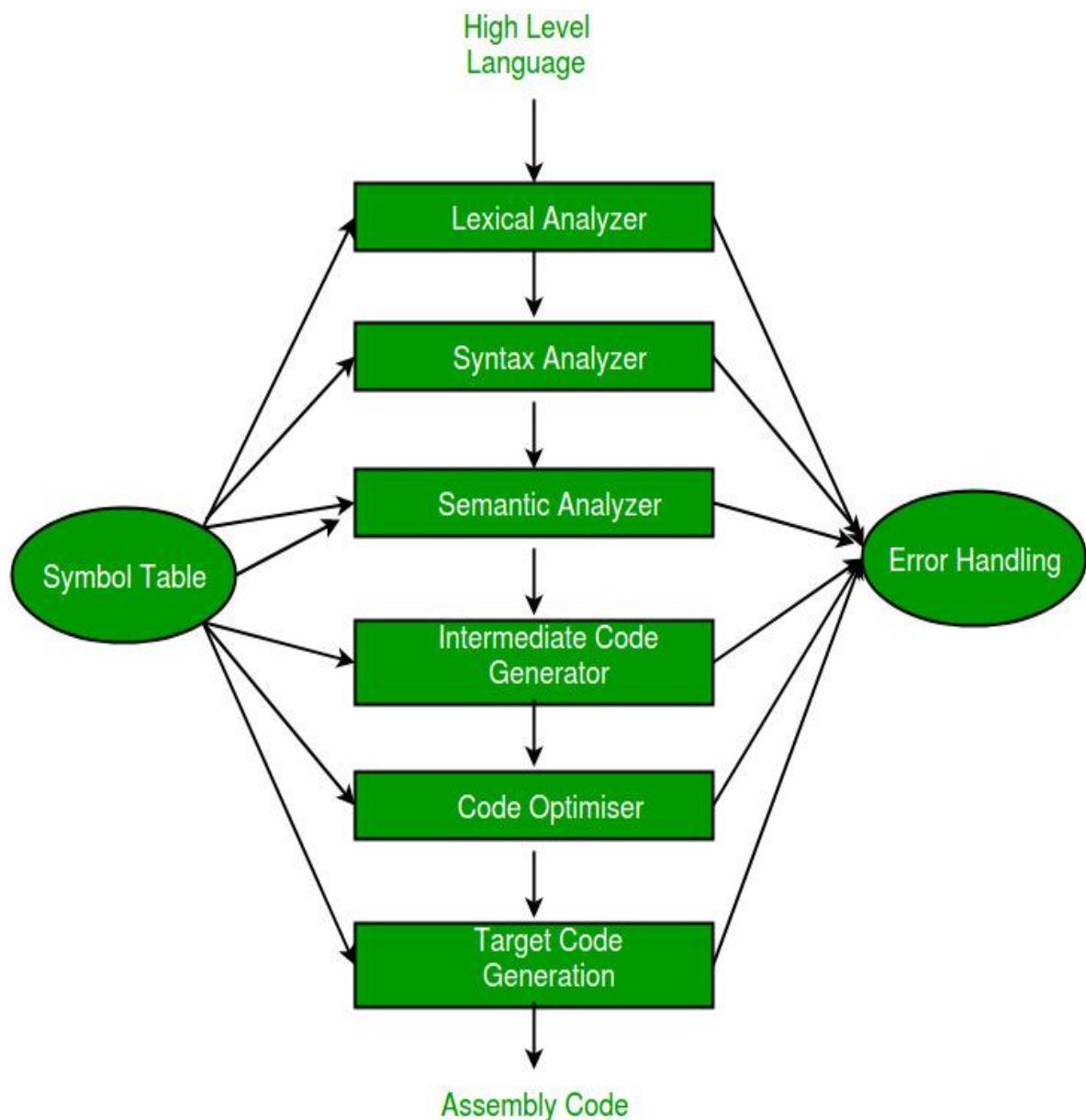
Compiler is an intelligent program as compare to an assembler. Compiler verifies all types of limits, ranges, errors, etc. Compiler program takes more time to run and it occupies huge amount of memory space. The speed of compiler is slower than other system software. It takes time because it enters through the program and then does translation of the full program. When compiler runs on same machine and produces machine code for the same machine on which it is running. Then it is called as self-compiler or resident compiler. Compiler may run on one machine and produces the machine codes for other computer then in that case it is called as cross compiler.

Stages of Compiler Design

- Lexical Analysis: The first stage of compiler design is lexical analysis, also known as scanning. In this stage, the compiler reads the source code character by character and breaks it down into a series of tokens, such as keywords, identifiers, and operators. These tokens are then passed on to the next stage of the compilation process.
- Syntax Analysis: The second stage of compiler design is syntax analysis, also known as parsing. In this stage, the compiler checks the syntax of the source code to ensure that it conforms to the rules of the programming language. The compiler builds a parse tree, which is a hierarchical representation of the program's structure, and uses it to check for syntax errors.
- Semantic Analysis: The third stage of compiler design is semantic analysis. In this stage, the compiler checks the meaning of the source code to ensure that it makes sense. The compiler performs type checking, which ensures that variables are used correctly and that operations are performed on compatible data types. The compiler also checks for other semantic errors, such as undeclared variables and incorrect function calls.
- Code Generation: The fourth stage of compiler design is code generation. In this stage, the compiler translates the parse tree into machine code that can be executed by the computer. The code generated by the compiler must be efficient and optimized for the target platform.
- Optimization: The final stage of compiler design is optimization. In this stage, the compiler analyses the generated code and makes optimizations to improve its performance. The compiler may perform optimizations such as constant folding, loop unrolling, and function inlining.

Overall, compiler design is a complex process that involves multiple stages and requires a deep understanding of both the programming language and the target platform. A well-designed compiler can

greatly improve the efficiency and performance of software programs, making them more useful and valuable for users.



PRACTICAL – 2

Write a program to check whether a string belong to the grammar or not.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main() {
    char string[50];
    int flag,count=0;
    printf("The grammar is:\n S->aS,\n S->Sb,\n S->ab\n");
    printf("Enter the string to be checked:\n");
    gets(string);
    if(string[0]=='a') {
        flag=0;
        for (count=1;string[count-1]!='\0';count++) {
            if(string[count]=='b') {
                flag=1;
                continue;
            }
            else if((flag==1)&&(string[count]=='a')) {
                printf("The string does not belong to the specified
grammar");
                break;
            }
            else if(string[count]=='a'){


```

```

        continue;

    }

else if((flag==1)&&(string[count]=='0')) {

    printf("String accepted.....!!!!");

    break;

}

else {

    printf("String not accepted");

}

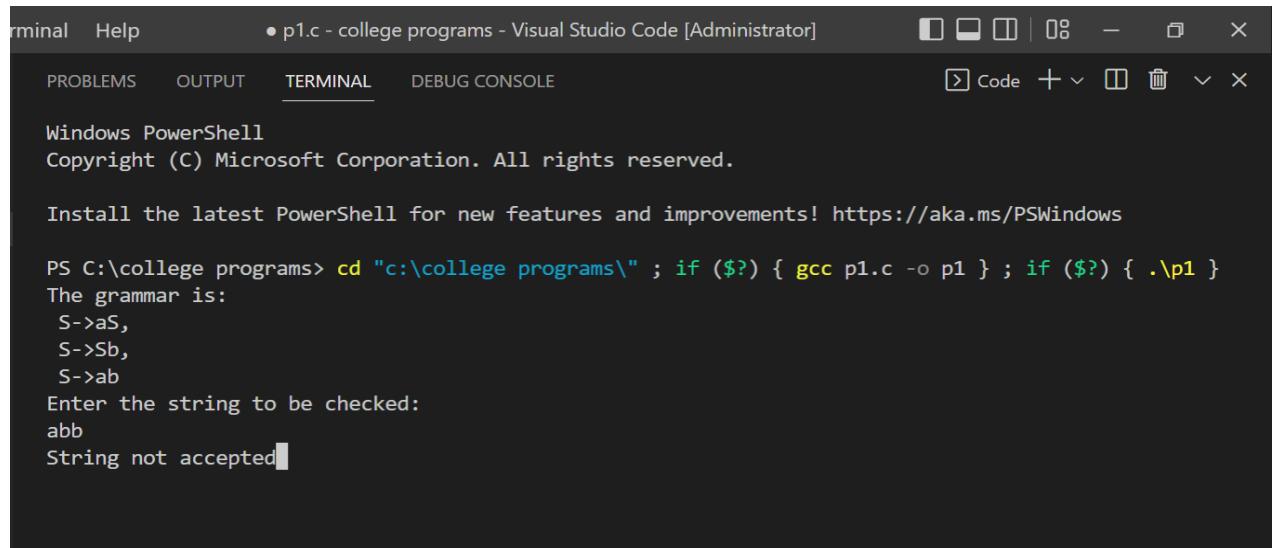
}

getch();

}

```

Output:



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal title is "p1.c - college programs - Visual Studio Code [Administrator]". The terminal content is as follows:

```

Terminal  Help          • p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE      Code  +  -  ×
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
The grammar is:
S->aS,
S->Sb,
S->ab
Enter the string to be checked:
abb
String not accepted

```

PRACTICAL – 3

Write a program to check whether a string include keyword or not.

Program:

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

bool isKeyword(char* str)
{
    if (!strcmp(str, "auto") || !strcmp(str, "default")
        || !strcmp(str, "signed") || !strcmp(str, "enum")
        || !strcmp(str, "extern") || !strcmp(str, "for")
        || !strcmp(str, "register") || !strcmp(str, "if")
        || !strcmp(str, "else") || !strcmp(str, "int")
        || !strcmp(str, "while") || !strcmp(str, "do")
        || !strcmp(str, "break") || !strcmp(str, "continue")
        || !strcmp(str, "double") || !strcmp(str, "float")
        || !strcmp(str, "return") || !strcmp(str, "char")
        || !strcmp(str, "case") || !strcmp(str, "const")
        || !strcmp(str, "sizeof") || !strcmp(str, "long")
        || !strcmp(str, "short") || !strcmp(str, "typedef")
        || !strcmp(str, "switch")
        || !strcmp(str, "unsigned") || !strcmp(str, "void")
        || !strcmp(str, "static") || !strcmp(str, "struct")
        || !strcmp(str, "goto") || !strcmp(str, "union")
        || !strcmp(str, "volatile")))
```

```
    return (true);

    return (false);

}

int main()

{

    printf("Is \"Geeks\" Keyword: \t");

    isKeyword("geeks") ? printf("Yes\n") : printf("No\n");

    printf("Is \"for\" Keyword: \t");

    isKeyword("for") ? printf("Yes\n") : printf("No\n");

    return 0;

}
```

Output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if (?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Is "Geeks" Keyword:      No
Is "for" Keyword:       Yes
PS C:\college programs>
```

PRACTICAL – 4

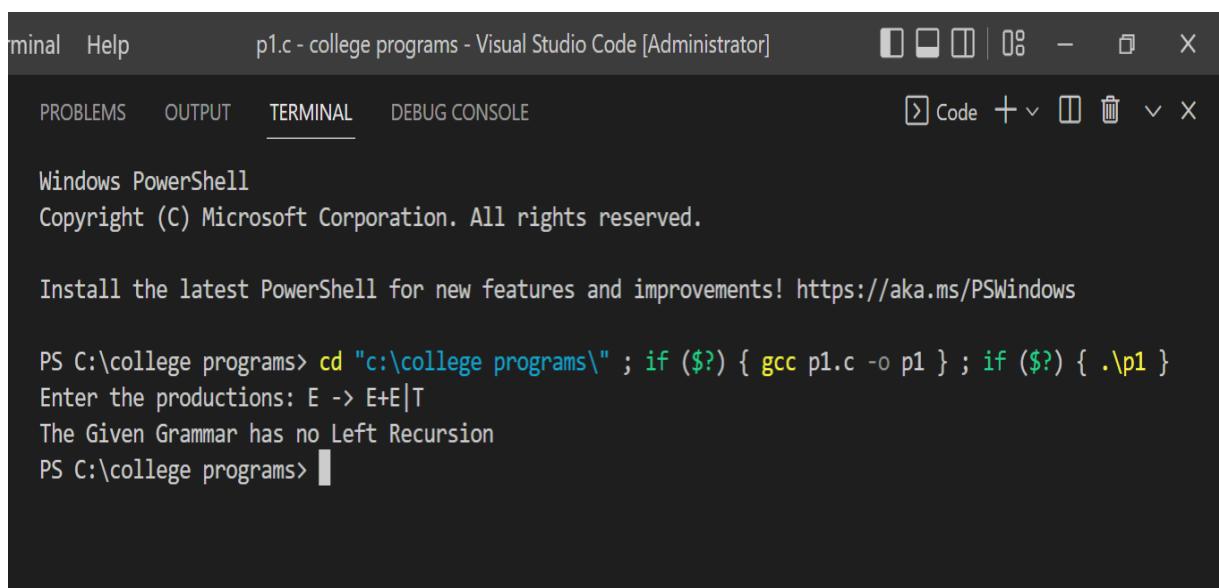
Write a program to remove left recursion from a grammar.

Program:

```
#include<stdio.h>
#include<string.h>
void main() {
    char input[100],l[50],r[50],temp[10],tempprod[20],productions[25][50];
    int i=0,j=0,flag=0,consumed=0;
    printf("Enter the productions: ");
    scanf("%1s->%s",l,r);
    printf("%s",r);
    while(sscanf(r+consumed,"%[^]s",temp) == 1 && consumed <= strlen(r)) {
        if(temp[0] == l[0]) {
            flag = 1;
            sprintf(productions[i++],"%s->%s%0s'\0",l,temp+1,l);
        }
        else
            sprintf(productions[i++],"%s'->%s%0s'\0",l,temp,l);
        consumed += strlen(temp)+1;
    }
    if(flag == 1) {
        sprintf(productions[i++],"%s->\0",l);
        printf("The productions after eliminating Left Recursion are:\n");
        for(j=0;j<i;j++)
            printf("%s\n",productions[j]);
    }
}
```

```
else
    printf("The Given Grammar has no Left Recursion");
}
```

Output:



A screenshot of the Visual Studio Code interface, specifically the terminal tab. The title bar shows "p1.c - college programs - Visual Studio Code [Administrator]". The terminal window displays the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
Enter the productions: E -> E+E|T
The Given Grammar has no Left Recursion
PS C:\college programs>
```

PRACTICAL – 5

Write a program to perform left factoring on a grammar.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
    char
gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[20];
    int i,j=0,k=0,l=0,pos;
    printf("Enter Production : A->");
    gets(gram);
    for(i=0;gram[i]!='|';i++,j++)
        part1[j]=gram[i];
    part1[j]='\0';
    for(j=++i,i=0;gram[j]!='\0';j++,i++)
        part2[i]=gram[j];
    part2[i]='\0';
    for(i=0;i<strlen(part1)||i<strlen(part2);i++){
        if(part1[i]==part2[i]){
            modifiedGram[k]=part1[i];
            k++;
            pos=i+1;
        }
    }
    for(i=pos,j=0;part1[i]!='\0';i++,j++){

```

```

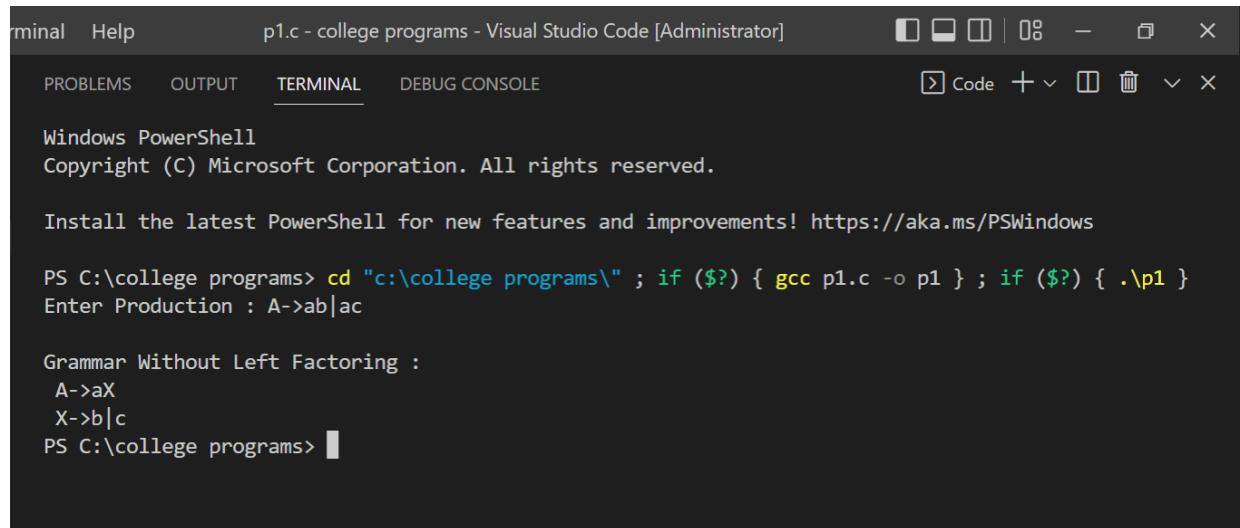
    newGram[j]=part1[i];
}

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

modifiedGram[k]='X';
modifiedGram[++k]='\0';
newGram[j]='\0';
printf("\nGrammar Without Left Factoring : \n");
printf(" A->%s",modifiedGram);
printf("\n X->%s\n",newGram);
}

```

Output:



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal tab is selected, showing the Windows PowerShell environment. The output displays the original grammar, its transformation into grammar without left factoring, and the resulting modified and new grammars.

```

Terminal Help p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Code + ×

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Enter Production : A->ab|ac

Grammar Without Left Factoring :
A->aX
X->b|c
PS C:\college programs>

```

PRACTICAL – 6

Write a program to show all the operations of a stack.

Program:

```
#include <stdio.h>
int MAXSIZE = 8;
int stack[8];
int top = -1;
int isempty(){
    if(top == -1)
        return 1;
    else
        return 0;
}
int isfull(){
    if(top == MAXSIZE)
        return 1;
    else
        return 0;
}
int peek(){
    return stack[top];
}
int pop(){
    int data;
    if(!isempty()) {
        data = stack[top];
        top--;
    }
    return data;
}
```

```

    top = top - 1;
    return data;
} else {
    printf("Could not retrieve data, Stack is empty.\n");
}
}

int push(int data){
    if(!isfull()) {
        top = top + 1;
        stack[top] = data;
    } else {
        printf("Could not insert data, Stack is full.\n");
    }
}

int main(){
    push(44);
    push(10);
    push(62);
    push(123);
    push(15);
    printf("Element at top of the stack: %d\n",peek());
    printf("Elements: \n");

    // print stack data
    while(!isempty()) {
        int data = pop();
        printf("%d\n",data);
    }
}

```

```
    }

    printf("Stack full: %s\n" , isfull()?"true":"false");

    printf("Stack empty: %s\n" , isempty()?"true":"false");

    return 0;
}


```

Output:

PRACTICAL – 7

Write a program to find out the leading of the non-terminals in a grammar.

Program:

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

char arr[18][3]={{'E', '+', 'F'}, {'E', '*', 'F'}, {'E', '(', 'F'}, {'E', ')', 'F'}, {'E', 'i', 'F'}, {'E', '$', 'F'},
{'F', '+', 'F'}, {'F', '*', 'F'}, {'F', '(', 'F'}, {'F', ')', 'F'}, {'F', 'i', 'F'}, {'F', '$', 'F'}, {'T', '+', 'F'},
{'T', '*', 'F'}, {'T', '(', 'F'}, {'T', ')', 'F'}, {'T', 'i', 'F'}, {'T', '$', 'F'}};
char prod[6] = "EETTFF";
char res[6][3]={{'E', '+', 'T'}, {'T', '\0'}, {'T', '*', 'F'}, {'F', '\0'}, {'(' , 'E', ')'}, {'i', '\0'}};
char stack [5][2];
int top = -1;

void install(char pro, char re) {
    int i;
    for (i = 0; i < 18; ++i) {
        if (arr[i][0] == pro && arr[i][1] == re) {
            arr[i][2] = 'T';
            break;
        }
    }
    ++top;
    stack[top][0] = pro;
```

```

stack[top][1] = re;
}

void main() {
    int i = 0, j;
    char pro, re, pri = ' ';
    for (i = 0; i < 6; ++i) {
        for (j = 0; j < 3 && res[i][j] != '\0'; ++j) {
            if (res[i][j] == '+' || res[i][j] == '*' || res[i][j] == '(' || res[i][j] == ')' ||
                res[i][j] == 'i' || res[i][j] == '$') {
                install(prod[i], res[i][j]);
                break;
            }
        }
    }
    while (top >= 0) {
        pro = stack[top][0];
        re = stack[top][1];
        --top;
        for (i = 0; i < 6; ++i) {
            if (res[i][0] == pro && res[i][0] != prod[i]) {
                install(prod[i], re);
            }
        }
    }
    for (i = 0; i < 18; ++i) {
        printf("\n\t");
        for (j = 0; j < 3; ++j)
            printf("%c\t", arr[i][j]);
    }
}

```

```

}

getch();

printf("\n\n");

for (i = 0; i < 18; ++i) {

    if (pri != arr[i][0]) {

        pri = arr[i][0];

        printf("\n\t%c -> ", pri);

    }

    if (arr[i][2] == 'T')

        printf("%c ", arr[i][1]);

    }

getch();

}

```

Output:

```

Terminal  Help      p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  Code  +  □  ▾  ×
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }

E      +
E      *
E      (
E      )
E      i
E      $
F      +
F      *
F      (
F      )
F      i
F      $
T      +
T      *
T      (
T      )
T      i
T      $

E -> + * ( i
F -> ( i
T -> * ( i
PS C:\college programs>

```

PRACTICAL – 8

Write a program to implement shift reduce parsing for a string.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
char ip_sym[15],stack[15];
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check();
void main()
{
    printf("\n\t\t SHIFT REDUCE PARSER\n");
    printf("\n GRAMMER\n");
    printf("\n E->E+E\n E->E/E");
    printf("\n E->E*E\n E->a/b");
    printf("\n enter the input symbol: ");
    gets(ip_sym);
    printf("\n\t stack implementation table");
    printf("\n stack \t\t input symbol\t\t action");
    printf("\n _____ \t\t _____ \t\t _____\n");
    printf("\n \$\t\t%$%\t\t-",ip_sym);
    strcpy(act,"shift ");
```

```

temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
len=strlen(ip_sym);
for(i=0;i<=len-1;i++)
{
stack[st_ptr]=ip_sym[ip_ptr];
stack[st_ptr+1]='\0';
ip_sym[ip_ptr]=' ';
ip_ptr++;
printf("\n $%s\t\t%s$\t\t%s",stack,ip_sym,act);
strcpy(act,"shift ");
temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
check();
st_ptr++;
}
st_ptr++;
check();
}

void check()
{
int flag=0;
temp2[0]=stack[st_ptr];
temp2[1]='\0';
if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))

```

```

{
stack[st_ptr]='E';
if(!strcmpi(temp2,"a"))
printf("\n $%s\t\t%s$\t\tE->a",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\tE->b",stack,ip_sym);
flag=1;
}
if((!strcmpi(temp2,"+"))||(!strcmpi(temp2,"*"))||(!strcmpi(temp2,"/"))) { flag=1;
}
if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E\E"))||(!strcmpi(stack,"E*E")))
{
strcpy(stack,"E");
st_ptr=0;
if(!strcmpi(stack,"E+E"))
printf("\n $%s\t\t%s$\t\tE->E+E",stack,ip_sym);
else
if(!strcmpi(stack,"E\E"))
printf("\n $%s\t\t%s$\t\tE->E\E",stack,ip_sym);
else if(!strcmpi(stack,"E*E"))
printf("\n $%s\t\t%s$\t\tE->E*E",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\tE->E+E",stack,ip_sym);
flag=1;
}
if(!strcmpi(stack,"E")&&ip_ptr==len)
{

```

```

printf("\n $%s\t\t%s$\t\tACCEPT",stack,ip_sym);

getch();

exit(0);

}

if(flag==0)

{

printf("\n%s\t\t%s$\t\t reject",stack,ip_sym);

exit(0);

}

return;
}

```

Output:

The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal title is "p1.c - college programs - Visual Studio Code [Administrator]". The terminal content displays the following output:

```

terminal Help p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
SHIFT REDUCE PARSER

GRAMMER

E->E+E
E->E/E
E->E*E
E->a/b
enter the input symbol: a+b

stack implementation table
-----      -----
stack      input symbol      action
-----      -----
$          a+b$              --
$a         +b$              shift a
$E         +b$              E->a
$E+
$E+b      b$              shift +
$E+E      $                shift b
$E         $                E->b
$E         $                E->E+E
$E         $                ACCEPT
PS C:\college programs>

```

PRACTICAL – 9

Write a program to find out the FIRST of the non-terminals in a grammar.

Program:

```
#include<stdio.h>
#include<conio.h>
char array[10][20],temp[10];
int c,n;
void fun(int,int[]);
int fun2(int i,int j,int p[],int );
void main()
{
    int p[2],i,j;
    printf("Enter the no. of productions :");
    scanf("%d",&n);
    printf("Enter the productions :\n");
    for(i=0;i<n;i++)
        scanf("%s",array[i]);
    for(i=0;i<n;i++)
    {
        c=-1,p[0]=-1,p[1]=-1;
        fun(i,p);
        printf("First(%c) : [ ",array[i][0]);
        for(j=0;j<=c;j++)
            printf("%c,",temp[j]);
        printf("\b ].\n");
    }
}
```

```

getch();
}

}

int fun2(int i,int j,int p[],int key)

{
    int k;
    if(!key)
    {
        for(k=0;k<n;k++)
        if(array[i][j]==array[k][0])
        break;
        p[0]=i;p[1]=j+1;
        fun(k,p);
        return 0;
    }
    else
    {
        for(k=0;k<=c;k++)
        {
            if(array[i][j]==temp[k])
            break;
        }
        if(k>c)return 1;
        else return 0;
    }
}
void fun(int i,int p[])

```

```

{
int j,k,key;
for(j=2;array[i][j] != NULL; j++)
{
if(array[i][j-1]=='/')
{
if(array[i][j]>= 'A' && array[i][j]<='Z')
{
key=0;
fun2(i,j,p,key);
}
else
{
key = 1;
if(fun2(i,j,p,key))
temp[++c] = array[i][j];
if(array[i][j]== '@'&& p[0]!=-1)
{
if(array[p[0]][p[1]]>='A' && array[p[0]][p[1]] <='Z')
{
key=0;
fun2(p[0],p[1],p,key);
}
else
if(array[p[0]][p[1]] != '/'&& array[p[0]][p[1]]!=NULL)
{
if(fun2(p[0],p[1],p,key))

```

```
temp[++c]=array[p[0]][p[1]];
}
}
}
}
}
}
```

Output:

```
Enter the no. of productions :6
Enter the productions :
S/aBDh
B/cC
C/bC/e
E/g/e
D/E/F
F/f/e
First(S) : [ a ].
First(B) : [ c ].
First(C) : [ b,e ].
First(E) : [ g,e ].
First(D) : [ g,e,f ].
First(F) : [ f,e ].
PS C:\college programs> █
```

PRACTICAL – 10

Write a program to check whether a grammar is operator
precedent.

Program:

```
#include<stdio.h>
#include<string.h>
char *input;
int i=0;
char lasthandle[6],stack[50],handles[][][5]={"")E","E*E","E+E","i","E^E"};
int top=0,l;
char prec[9][9]={
    '>','>','<','<','<','<','<','<','>','>',
    '>','>','<','<','<','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','>','e','e','>','>',
    '<','<','<','<','<','<','<','<','>','e',
    '>','>','>','>','>','e','e','>','>',
    '<','<','<','<','<','<','<','<','>','>',
    };
int getindex(char c)
{
switch(c)
{
case '+':return 0;
```

```

        case '-':return 1;
        case '*':return 2;
        case '/':return 3;
        case '^':return 4;
        case 'i':return 5;
        case '(':return 6;
        case ')':return 7;
        case '$':return 8;
    }
}

int shift()
{
    stack[++top]=*(input+i++);
    stack[top+1]='\0';
}

int reduce()
{
    int i,len,found,t;
    for(i=0;i<5;i++)
    {
        len=strlen(handles[i]);
        if(stack[top]==handles[i][0]&&top+1>=len)
        {
            found=1;
            for(t=0;t<len;t++)
            {
                if(stack[top-t]!=handles[i][t])

```

```

    {
        found=0;
        break;
    }

    if(found==1)
    {
        stack[top-t+1]='E';
        top=top-t+1;
        strcpy(lasthandle,handles[i]);
        stack[top+1]='\0';
        return 1;
    }
}

return 0;
}

void dispstack()
{
    int j;
    for(j=0;j<=top;j++)
        printf("%c",stack[j]);
}

void dispinput()
{
    int j;
    for(j=i;j<l;j++)

```

```

    printf("%c",*(input+j));
}

void main()
{
int j;
input=(char*)malloc(50*sizeof(char));
printf("\nEnter the string\n");
scanf("%s",input);
input=strcat(input,"$");
l=strlen(input);
strcpy(stack,"$");
printf("\nSTACK\tINPUT\tACTION");
while(i<=l)
{
    shift();
    printf("\n");
    dispstack();
    printf("\t");
    dispinput();
    printf("\tShift");
    if(prec[getindex(stack[top])][getindex(input[i])]=='>
')
{
    while(reduce())
    {
        printf("\n");
        dispstack();
        printf("\t");
    }
}
}

```

```

        dispinput();
        printf("\tReduced: E->%s",lasthandle);
    }
}

if(strcmp(stack,"$E$")==0)
    printf("\nAccepted;");
else
    printf("\nNot Accepted;");
}

```

Output:

```

Enter the string
i*(i+i)*i

STACK   INPUT   ACTION
$i      *(i+i)*i$     Shift
$E      *(i+i)*i$     Reduced: E->i
$E*     (i+i)*i$     Shift
$E*(    i+i)*i$ Shift
$E*(i   +i)*i$ Shift
$E*(E   +i)*i$ Reduced: E->i
$E*(E+  i)*i$ Shift
$E*(E+i )*i$ Shift
$E*(E+E )*i$ Reduced: E->i
$E*(E   )*i$ Reduced: E->E+E
$E*(E)  *i$ Shift
$E*E   *i$ Reduced: E->)E(
$E   *i$ Reduced: E->E*E
$E*   i$ Shift
$E*i   $ Shift
$E*E   $ Reduced: E->i
$E   $ Reduced: E->E*E
$E$    Shift
$E$    Shift
Accepted;
PS C:\college programs> █

```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
P  BT  WT  TAT
1 10  0   10
2 5   10  15
3 8   15  23
Average waiting time = 8.333333
Average turn around time = 16.000000
PS C:\college programs>

```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
Enter number of process: 5
Enter Burst Time:
P1: 01
P2: 02
P3: 03
P4: 04
P5: 05
P      BT      WT      TAT
P1     1       0       1
P2     2       1       3
P3     3       3       6
P4     4       6      10
P5     5      10      15
Average Waiting Time= 4.000000
Average Turnaround Time= 7.000000
PS C:\college programs>

```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
Enter Number of Processes: 5
Enter Burst Time and Priority Value for Process 1: 1 3
Enter Burst Time and Priority Value for Process 2: 2 2
Enter Burst Time and Priority Value for Process 3: 3 1
Enter Burst Time and Priority Value for Process 4: 4 5
Enter Burst Time and Priority Value for Process 5: 5 4
Order of process Execution is
P4 is executed from 0 to 4
P5 is executed from 4 to 9
P1 is executed from 9 to 10
P2 is executed from 10 to 12
P3 is executed from 12 to 15

Process Id    Burst Time   Wait Time   TurnAround Time
P4            4             0           4
P5            5             4           9
P1            1             9          10
P2            2            10          12
P3            3            12          15
PS C:\college programs>

```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Total number of process in the system: 2

Enter the Arrival and Burst time of the Process[1]
Arrival time is: 1

Burst time is: 8

Enter the Arrival and Burst time of the Process[2]
Arrival time is: 2

Burst time is: 4
Enter the Time Quantum for the process: 3

Process No          Burst Time          TAT          Waiting Time
Process No[2]        4                  8           4
Process No[1]        8                  11          3
Average Turn Around Time: 9.500000
Average Waiting Time: 3.500000

```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Enter no of pages:10
Enter the reference string:7 8 9 3 2 4 7 4 6 1
Enter no of frames:3

7
7     8
7     8     9
3     8     9
3     2     9
3     2     4
7     2     4
7     6     4
1     6     4

The no of page faults is 9
PS C:\college programs>

```

Incoming	t	Frame 1	t	Frame 2	t	Frame 3	
4		4		-		-	
1		4		1		-	
2		4		1		2	
4		4		1		2	
5		5		1		2	
Total Page Faults: 4							

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Enter number of frames: 3
Enter number of pages: 10
Enter page reference string: 7 8 9 3 2 1 6 5 4 9

7      -1      -1
7      8      -1
7      8      9
3      8      9
2      8      9
1      8      9
6      8      9
5      8      9
4      8      9
4      8      9

Total Page Faults = 9
PS C:\college programs>

```

```
Memory Management Scheme - Worst Fit
Enter the number of blocks:3
Enter the number of files:3

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4
File 3:5

File_no:      File_size :      Block_no:      Block_size:      Fragement
1            1              3                7                  6
2            4              1                5                  1
3            5              0                8                  0
```

```
Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File_no:      File_size :      Block_no:      Block_size:      Fragement
1            1              1                5                  4
2            4              3                7                  3
```

```
PS C:\college programs> █

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:7
Block 3:2
Enter the size of the files :-
File 1:4
File 2:3

File No File Size      Block No      Block Size      Fragment
1          4             1              5                  1
2          3             2              7                  4
```

```
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving
1 reader is inside
1 Reader is leaving
Writer is trying to enter
Writer has entered
Writer is leaving

...Program finished with exit code 0
Press ENTER to exit console. █
```

```
Enter the number of Producers:2
Enter the number of Consumers:2
Enter buffer capacity:2
Successfully created producer 1
Producer 1 produced 29
Successfully created producer 2
Producer 2 produced 11
Successfully created consumer 1
Buffer:29 11
Consumer 3 consumed 11
Current buffer len: 1
Buffer:29
Consumer 2 consumed 29
Current buffer len: 0
Successfully created consumer 2
Producer 1 produced 3
Buffer:3
Consumer 2 consumed 3
Current buffer len: 0
Producer 2 produced 31
Producer 1 produced 27
Producer 1 produced 7
Buffer:31 27
```

```
Enter the number of process and resources
5 3
enter allocation of resource of all process 5x3 matrix
1 2 3
4 5 6
0 6 7
0 8 9
9 5 4
enter the max resource process required 5x3 matrix
0 1 0
1 2 3
2 3 4
3 4 5
4 5 6
enter the available resource 3 4 2

need resources matrix are
-1      -1      -3
-3      -3      -3
2       -3      -3
3       -4      -4
-5      0       2

available resource after completion
17      30      31
safe sequence are
p0      p1      p2      p3      p4
```

```
Enter the directory name:siddhika
Enter the number of files:3
Enter file name to be created:sid
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:shukla
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:ss
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:sk
Do you want to enter another file(yes - 1 or no - 0):0
Directory name is:siddhika
Files names are:
sid
shukla
ss
sk
PS C:\college programs> █
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 1
```

```
Enter name of directory -- siddhika
Directory created
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 2
```

```
Enter name of the directory -- siddhika
Enter name of the file -- sid
File created
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 5
```

```
Directory      Files
siddhika      sid
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 6
PS C:\college programs> █
```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc hierarchical.c -o hierarchical } ; if (?) { .\hierarchical }
Enter number of users: 1
Enter name: siddhika
Enter dir(1) or file(0): sid
```

```
Hierarchical
█
```

GURU TEGH BAHADUR INSTITUTE OF
TECHNOLOGY

COMPILER DESIGN LAB FILE

NAME: SIDDHIKA SHUKLA

ENROLLMENT NO.: 05113203121

BRANCH: IT-1, 5TH SEM

TITLE TABLE

PRACTICAL – 1

Introduction to Compiler Design.

The compiler is software that converts a program written in a high-level language (Source Language) to a low-level language (Object/Target/Machine Language/0, 1's).

A translator or language processor is a program that translates an input program written in a programming language into an equivalent program in another language. The compiler is a type of translator, which takes a program written in a high-level programming language as input and translates it into an equivalent program in low-level languages such as machine language or assembly language.

The program written in a high-level language is known as a source program, and the program converted into a low-level language is known as an object (or target) program. Without compilation, no program written in a high-level language can be executed. For every programming language, we have a different compiler; however, the basic tasks performed by every compiler are the same. The process of translating the source code into machine code involves several stages, including lexical analysis, syntax analysis, semantic analysis, code generation, and optimization.

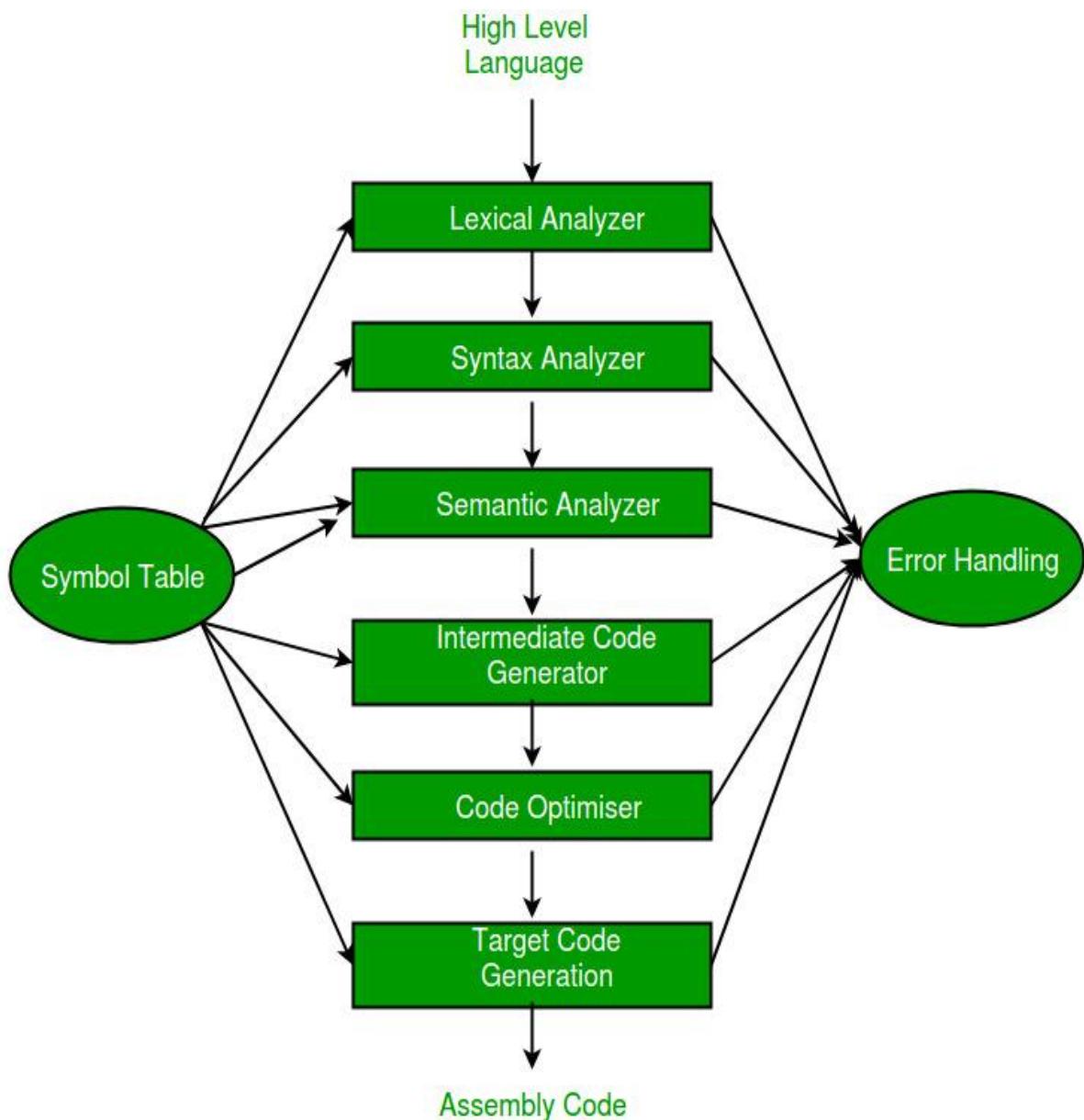
Compiler is an intelligent program as compare to an assembler. Compiler verifies all types of limits, ranges, errors, etc. Compiler program takes more time to run and it occupies huge amount of memory space. The speed of compiler is slower than other system software. It takes time because it enters through the program and then does translation of the full program. When compiler runs on same machine and produces machine code for the same machine on which it is running. Then it is called as self-compiler or resident compiler. Compiler may run on one machine and produces the machine codes for other computer then in that case it is called as cross compiler.

Stages of Compiler Design

- Lexical Analysis: The first stage of compiler design is lexical analysis, also known as scanning. In this stage, the compiler reads the source code character by character and breaks it down into a series of tokens, such as keywords, identifiers, and operators. These tokens are then passed on to the next stage of the compilation process.
- Syntax Analysis: The second stage of compiler design is syntax analysis, also known as parsing. In this stage, the compiler checks the syntax of the source code to ensure that it conforms to the rules of the programming language. The compiler builds a parse tree, which is a hierarchical representation of the program's structure, and uses it to check for syntax errors.
- Semantic Analysis: The third stage of compiler design is semantic analysis. In this stage, the compiler checks the meaning of the source code to ensure that it makes sense. The compiler performs type checking, which ensures that variables are used correctly and that operations are performed on compatible data types. The compiler also checks for other semantic errors, such as undeclared variables and incorrect function calls.
- Code Generation: The fourth stage of compiler design is code generation. In this stage, the compiler translates the parse tree into machine code that can be executed by the computer. The code generated by the compiler must be efficient and optimized for the target platform.
- Optimization: The final stage of compiler design is optimization. In this stage, the compiler analyses the generated code and makes optimizations to improve its performance. The compiler may perform optimizations such as constant folding, loop unrolling, and function inlining.

Overall, compiler design is a complex process that involves multiple stages and requires a deep understanding of both the programming language and the target platform. A well-designed compiler can

greatly improve the efficiency and performance of software programs, making them more useful and valuable for users.



PRACTICAL – 2

Write a program to check whether a string belong to the grammar or not.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main() {
    char string[50];
    int flag,count=0;
    printf("The grammar is:\n S->aS,\n S->Sb,\n S->ab\n");
    printf("Enter the string to be checked:\n");
    gets(string);
    if(string[0]=='a') {
        flag=0;
        for (count=1;string[count-1]!='\0';count++) {
            if(string[count]=='b') {
                flag=1;
                continue;
            }
            else if((flag==1)&&(string[count]=='a')) {
                printf("The string does not belong to the specified
grammar");
                break;
            }
            else if(string[count]=='a'){


```

```

        continue;

    }

else if((flag==1)&&(string[count]=='0')) {

    printf("String accepted.....!!!!");

    break;

}

else {

    printf("String not accepted");

}

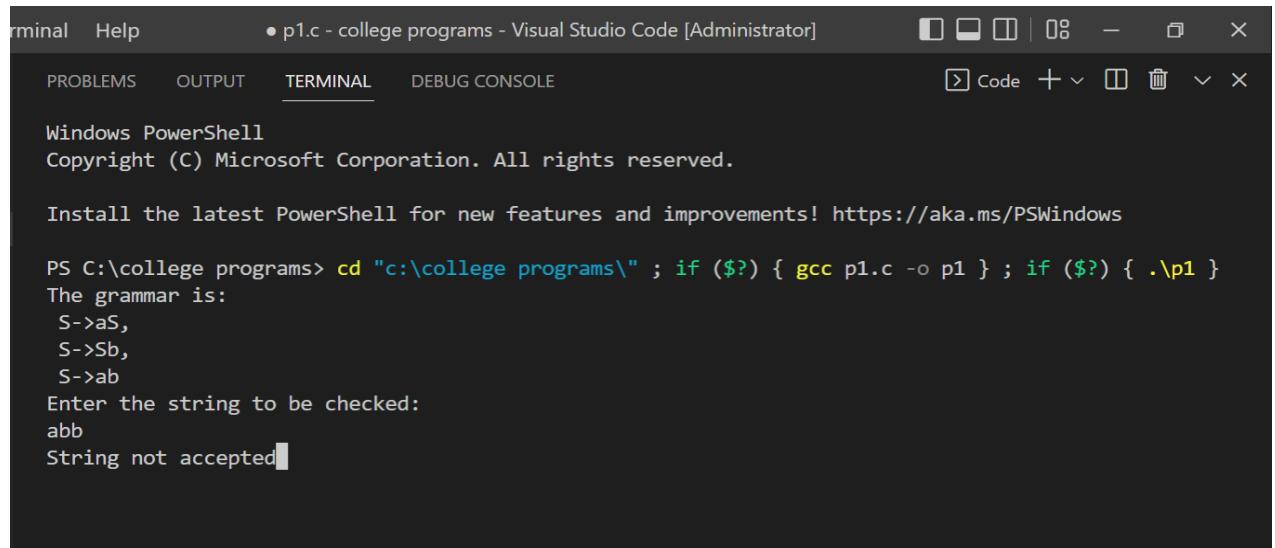
}

getch();

}

```

Output:



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal tab is selected, showing the output of a C program. The program defines a grammar S -> aS, S -> Sb, and S -> ab. It then prompts the user to enter a string to be checked. The user enters 'abb', and the program outputs 'String not accepted'.

```

Terminal  Help      • p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE   Code + ×   Code + ×
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
The grammar is:
S->aS,
S->Sb,
S->ab
Enter the string to be checked:
abb
String not accepted

```

PRACTICAL – 3

Write a program to check whether a string include keyword or not.

Program:

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

bool isKeyword(char* str)
{
    if (!strcmp(str, "auto") || !strcmp(str, "default")
        || !strcmp(str, "signed") || !strcmp(str, "enum")
        || !strcmp(str, "extern") || !strcmp(str, "for")
        || !strcmp(str, "register") || !strcmp(str, "if")
        || !strcmp(str, "else") || !strcmp(str, "int")
        || !strcmp(str, "while") || !strcmp(str, "do")
        || !strcmp(str, "break") || !strcmp(str, "continue")
        || !strcmp(str, "double") || !strcmp(str, "float")
        || !strcmp(str, "return") || !strcmp(str, "char")
        || !strcmp(str, "case") || !strcmp(str, "const")
        || !strcmp(str, "sizeof") || !strcmp(str, "long")
        || !strcmp(str, "short") || !strcmp(str, "typedef")
        || !strcmp(str, "switch")
        || !strcmp(str, "unsigned") || !strcmp(str, "void")
        || !strcmp(str, "static") || !strcmp(str, "struct")
        || !strcmp(str, "goto") || !strcmp(str, "union")
        || !strcmp(str, "volatile")))
```

```
    return (true);

    return (false);

}

int main()

{

    printf("Is \"Geeks\" Keyword: \t");

    isKeyword("geeks") ? printf("Yes\n") : printf("No\n");

    printf("Is \"for\" Keyword: \t");

    isKeyword("for") ? printf("Yes\n") : printf("No\n");

    return 0;

}
```

Output:

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

PS C:\college programs> cd "c:\college programs\" ; if (\$?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Is "Geeks" Keyword: No
Is "for" Keyword: Yes
PS C:\college programs>

PRACTICAL – 4

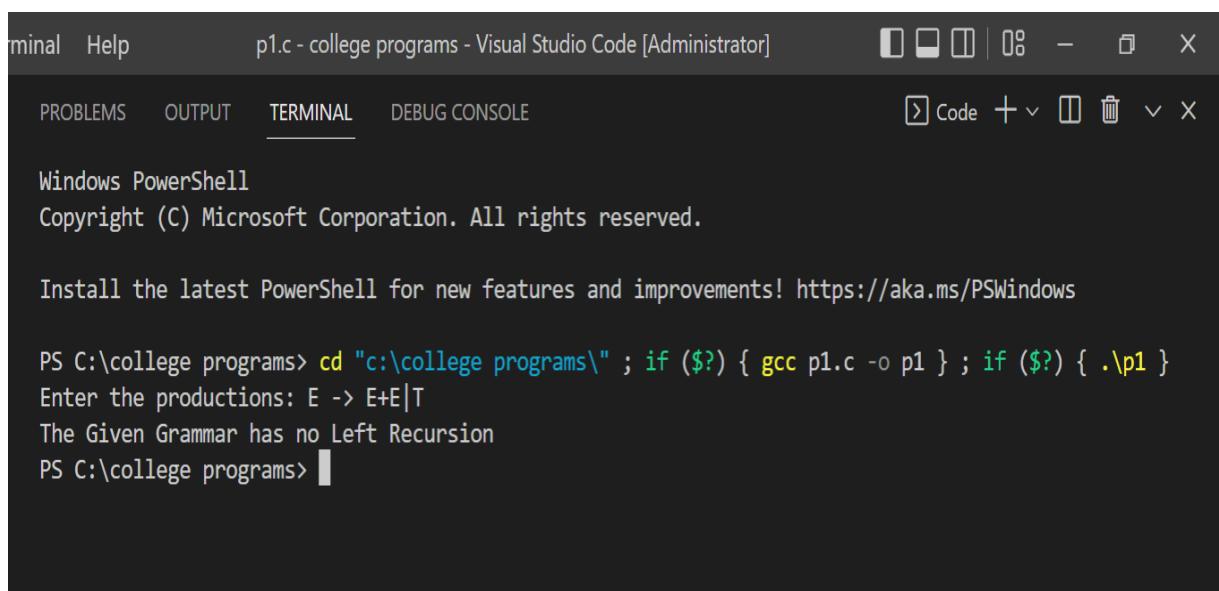
Write a program to remove left recursion from a grammar.

Program:

```
#include<stdio.h>
#include<string.h>
void main() {
    char input[100],l[50],r[50],temp[10],tempprod[20],productions[25][50];
    int i=0,j=0,flag=0,consumed=0;
    printf("Enter the productions: ");
    scanf("%1s->%s",l,r);
    printf("%s",r);
    while(sscanf(r+consumed,"%[^]s",temp) == 1 && consumed <= strlen(r)) {
        if(temp[0] == l[0]) {
            flag = 1;
            sprintf(productions[i++],"%s->%s%0s'\0",l,temp+1,l);
        }
        else
            sprintf(productions[i++],"%s'->%s%0s'\0",l,temp,l);
        consumed += strlen(temp)+1;
    }
    if(flag == 1) {
        sprintf(productions[i++],"%s->\0",l);
        printf("The productions after eliminating Left Recursion are:\n");
        for(j=0;j<i;j++)
            printf("%s\n",productions[j]);
    }
}
```

```
else
    printf("The Given Grammar has no Left Recursion");
}
```

Output:



A screenshot of the Visual Studio Code interface, specifically the terminal tab. The title bar shows "p1.c - college programs - Visual Studio Code [Administrator]". The terminal window displays the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
Enter the productions: E -> E+E|T
The Given Grammar has no Left Recursion
PS C:\college programs>
```

PRACTICAL – 5

Write a program to perform left factoring on a grammar.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
    char
gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[20];
    int i,j=0,k=0,l=0,pos;
    printf("Enter Production : A->");
    gets(gram);
    for(i=0;gram[i]!='|';i++,j++)
        part1[j]=gram[i];
    part1[j]='\0';
    for(j=++i,i=0;gram[j]!='\0';j++,i++)
        part2[i]=gram[j];
    part2[i]='\0';
    for(i=0;i<strlen(part1)||i<strlen(part2);i++){
        if(part1[i]==part2[i]){
            modifiedGram[k]=part1[i];
            k++;
            pos=i+1;
        }
    }
    for(i=pos,j=0;part1[i]!='\0';i++,j++){

```

```

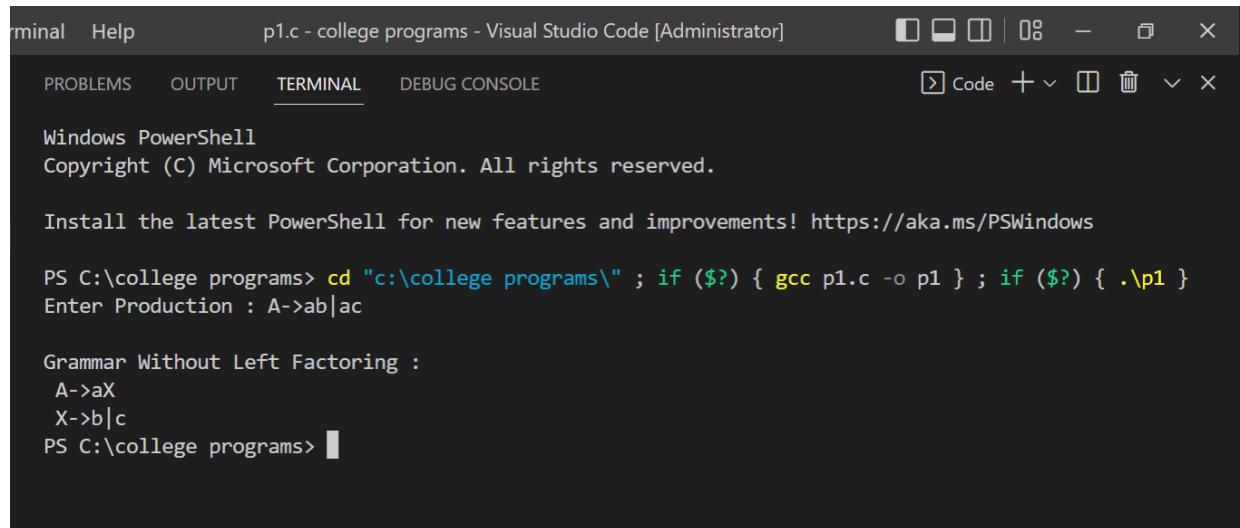
    newGram[j]=part1[i];
}

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

modifiedGram[k]='X';
modifiedGram[++k]='\0';
newGram[j]='\0';
printf("\nGrammar Without Left Factoring : \n");
printf(" A->%s",modifiedGram);
printf("\n X->%s\n",newGram);
}

```

Output:



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal tab is selected, showing the Windows PowerShell environment. The output displays the original grammar rules, the transformed grammar without left factoring, and the resulting modified and new grammars.

```

Terminal Help p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Code + ×

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Enter Production : A->ab|ac

Grammar Without Left Factoring :
A->aX
X->b|c
PS C:\college programs>

```

PRACTICAL – 6

Write a program to show all the operations of a stack.

Program:

```
#include <stdio.h>
int MAXSIZE = 8;
int stack[8];
int top = -1;
int isempty(){
    if(top == -1)
        return 1;
    else
        return 0;
}
int isfull(){
    if(top == MAXSIZE)
        return 1;
    else
        return 0;
}
int peek(){
    return stack[top];
}
int pop(){
    int data;
    if(!isempty()) {
        data = stack[top];
        top--;
    }
    return data;
}
```

```

    top = top - 1;
    return data;
} else {
    printf("Could not retrieve data, Stack is empty.\n");
}
}

int push(int data){
    if(!isfull()) {
        top = top + 1;
        stack[top] = data;
    } else {
        printf("Could not insert data, Stack is full.\n");
    }
}

int main(){
    push(44);
    push(10);
    push(62);
    push(123);
    push(15);
    printf("Element at top of the stack: %d\n",peek());
    printf("Elements: \n");

    // print stack data
    while(!isempty()) {
        int data = pop();
        printf("%d\n",data);
    }
}

```

```

    }

printf("Stack full: %s\n" , isfull()?"true":"false");

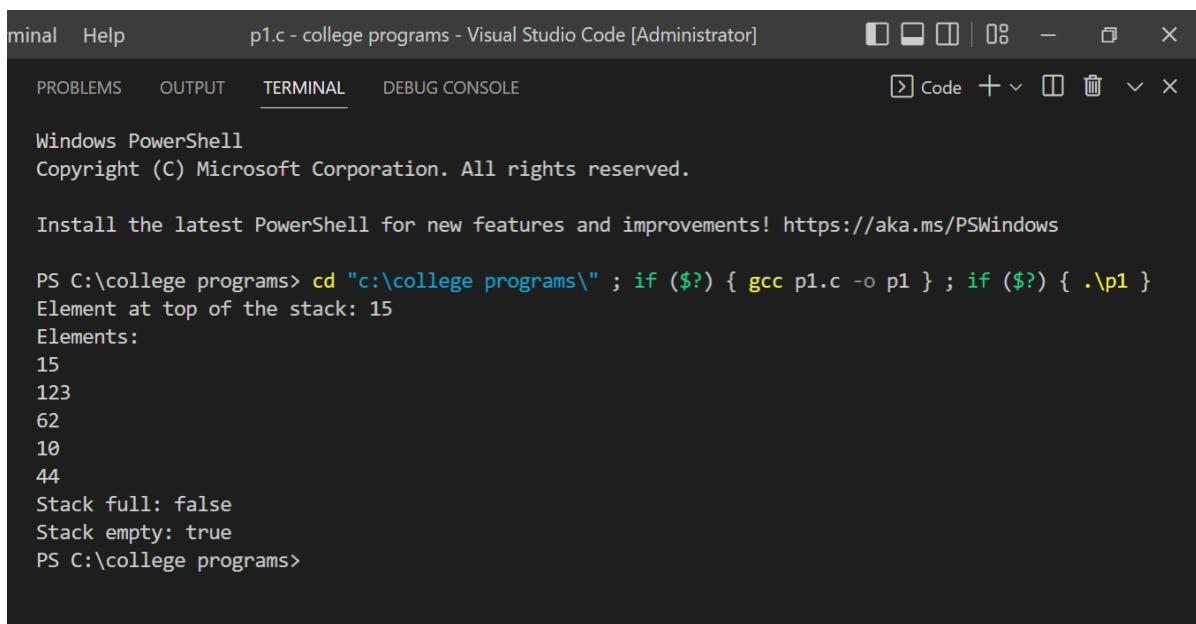
printf("Stack empty: %s\n" , isempty()?"true":"false");

return 0;

}

```

Output:



The screenshot shows a Visual Studio Code interface with a terminal window open. The title bar reads "p1.c - college programs - Visual Studio Code [Administrator]". The terminal tab is active, displaying the following output:

```

minal  Help          p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  Code  +  □  □  ▾  X
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
Element at top of the stack: 15
Elements:
15
123
62
10
44
Stack full: false
Stack empty: true
PS C:\college programs>

```

PRACTICAL – 7

Write a program to find out the leading of the non-terminals in a grammar.

Program:

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

char arr[18][3]={{'E', '+', 'F'}, {'E', '*', 'F'}, {'E', '(', 'F'}, {'E', ')', 'F'}, {'E', 'i', 'F'}, {'E', '$', 'F'},
{'F', '+', 'F'}, {'F', '*', 'F'}, {'F', '(', 'F'}, {'F', ')', 'F'}, {'F', 'i', 'F'}, {'F', '$', 'F'}, {'T', '+', 'F'},
{'T', '*', 'F'}, {'T', '(', 'F'}, {'T', ')', 'F'}, {'T', 'i', 'F'}, {'T', '$', 'F'}};
char prod[6] = "EETTFF";
char res[6][3]={{'E', '+', 'T'}, {'T', '\0'}, {'T', '*', 'F'}, {'F', '\0'}, {'(' , 'E', ')'}, {'i', '\0'}};
char stack [5][2];
int top = -1;

void install(char pro, char re) {
    int i;
    for (i = 0; i < 18; ++i) {
        if (arr[i][0] == pro && arr[i][1] == re) {
            arr[i][2] = 'T';
            break;
        }
    }
    ++top;
    stack[top][0] = pro;
```

```

stack[top][1] = re;
}

void main() {
    int i = 0, j;
    char pro, re, pri = ' ';
    for (i = 0; i < 6; ++i) {
        for (j = 0; j < 3 && res[i][j] != '\0'; ++j) {
            if (res[i][j] == '+' || res[i][j] == '*' || res[i][j] == '(' || res[i][j] == ')' ||
                res[i][j] == 'i' || res[i][j] == '$') {
                install(prod[i], res[i][j]);
                break;
            }
        }
    }
    while (top >= 0) {
        pro = stack[top][0];
        re = stack[top][1];
        --top;
        for (i = 0; i < 6; ++i) {
            if (res[i][0] == pro && res[i][0] != prod[i]) {
                install(prod[i], re);
            }
        }
    }
    for (i = 0; i < 18; ++i) {
        printf("\n\t");
        for (j = 0; j < 3; ++j)
            printf("%c\t", arr[i][j]);
    }
}

```

```

}

getch();

printf("\n\n");

for (i = 0; i < 18; ++i) {

    if (pri != arr[i][0]) {

        pri = arr[i][0];

        printf("\n\t%c -> ", pri);

    }

    if (arr[i][2] == 'T')

        printf("%c ", arr[i][1]);

    }

getch();

}

```

Output:

```

Terminal  Help      p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  Code  +  □  ▾  ×
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }

E      +
E      *
E      (
E      )
E      i
E      $
F      +
F      *
F      (
F      )
F      i
F      $
T      +
T      *
T      (
T      )
T      i
T      $

E -> + * ( i
F -> ( i
T -> * ( i
PS C:\college programs>

```

PRACTICAL – 8

Write a program to implement shift reduce parsing for a string.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
char ip_sym[15],stack[15];
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check();
void main()
{
    printf("\n\t\t SHIFT REDUCE PARSER\n");
    printf("\n GRAMMER\n");
    printf("\n E->E+E\n E->E/E");
    printf("\n E->E*E\n E->a/b");
    printf("\n enter the input symbol: ");
    gets(ip_sym);
    printf("\n\t stack implementation table");
    printf("\n stack \t\t input symbol\t\t action");
    printf("\n _____ \t\t _____ \t\t _____\n");
    printf("\n \$\t\t%$%\t\t-",ip_sym);
    strcpy(act,"shift ");
```

```

temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
len=strlen(ip_sym);
for(i=0;i<=len-1;i++)
{
stack[st_ptr]=ip_sym[ip_ptr];
stack[st_ptr+1]='\0';
ip_sym[ip_ptr]=' ';
ip_ptr++;
printf("\n $%s\t\t%s$\t\t%s",stack,ip_sym,act);
strcpy(act,"shift ");
temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
check();
st_ptr++;
}
st_ptr++;
check();
}

void check()
{
int flag=0;
temp2[0]=stack[st_ptr];
temp2[1]='\0';
if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))

```

```

{
stack[st_ptr]='E';
if(!strcmpi(temp2,"a"))
printf("\n $%s\t\t%s$\t\tE->a",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\tE->b",stack,ip_sym);
flag=1;
}
if((!strcmpi(temp2,"+"))||(!strcmpi(temp2,"*"))||(!strcmpi(temp2,"/"))) { flag=1;
}
if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E\E"))||(!strcmpi(stack,"E*E")))
{
strcpy(stack,"E");
st_ptr=0;
if(!strcmpi(stack,"E+E"))
printf("\n $%s\t\t%s$\t\tE->E+E",stack,ip_sym);
else
if(!strcmpi(stack,"E\E"))
printf("\n $%s\t\t%s$\t\tE->E\E",stack,ip_sym);
else if(!strcmpi(stack,"E*E"))
printf("\n $%s\t\t%s$\t\tE->E*E",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\tE->E+E",stack,ip_sym);
flag=1;
}
if(!strcmpi(stack,"E")&&ip_ptr==len)
{

```

```

printf("\n $%s\t\t%s$\t\tACCEPT",stack,ip_sym);

getch();

exit(0);

}

if(flag==0)

{

printf("\n%s\t\t%s$\t\t reject",stack,ip_sym);

exit(0);

}

return;
}

```

Output:

The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal title is "p1.c - college programs - Visual Studio Code [Administrator]". The terminal content displays the following output:

```

terminal Help p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
SHIFT REDUCE PARSER

GRAMMER

E->E+E
E->E/E
E->E*E
E->a/b
enter the input symbol: a+b

stack implementation table
-----      -----
stack      input symbol      action
-----      -----
$          a+b$              --
$a         +b$              shift a
$E         +b$              E->a
$E+
$E+b      b$              shift +
$E+E      $                shift b
$E         $                E->b
$E         $                E->E+E
$E         $                ACCEPT
PS C:\college programs>

```

PRACTICAL – 9

Write a program to find out the FIRST of the non-terminals in a grammar.

Program:

```
#include<stdio.h>
#include<conio.h>
char array[10][20],temp[10];
int c,n;
void fun(int,int[]);
int fun2(int i,int j,int p[],int );
void main()
{
    int p[2],i,j;
    printf("Enter the no. of productions :");
    scanf("%d",&n);
    printf("Enter the productions :\n");
    for(i=0;i<n;i++)
        scanf("%s",array[i]);
    for(i=0;i<n;i++)
    {
        c=-1,p[0]=-1,p[1]=-1;
        fun(i,p);
        printf("First(%c) : [ ",array[i][0]);
        for(j=0;j<=c;j++)
            printf("%c,",temp[j]);
        printf("\b ].\n");
    }
}
```

```

getch();
}

}

int fun2(int i,int j,int p[],int key)

{
    int k;
    if(!key)
    {
        for(k=0;k<n;k++)
        if(array[i][j]==array[k][0])
        break;
        p[0]=i;p[1]=j+1;
        fun(k,p);
        return 0;
    }
    else
    {
        for(k=0;k<=c;k++)
        {
            if(array[i][j]==temp[k])
            break;
        }
        if(k>c)return 1;
        else return 0;
    }
}
void fun(int i,int p[])

```

```

{
int j,k,key;
for(j=2;array[i][j] != NULL; j++)
{
if(array[i][j-1]=='/')
{
if(array[i][j]>= 'A' && array[i][j]<='Z')
{
key=0;
fun2(i,j,p,key);
}
else
{
key = 1;
if(fun2(i,j,p,key))
temp[++c] = array[i][j];
if(array[i][j]== '@'&& p[0]!=-1)
{
if(array[p[0]][p[1]]>='A' && array[p[0]][p[1]] <='Z')
{
key=0;
fun2(p[0],p[1],p,key);
}
else
if(array[p[0]][p[1]] != '/'&& array[p[0]][p[1]]!=NULL)
{
if(fun2(p[0],p[1],p,key))

```

```
temp[++c]=array[p[0]][p[1]];
}
}
}
}
}
}
```

Output:

```
Enter the no. of productions :6
Enter the productions :
S/aBDh
B/cC
C/bC/e
E/g/e
D/E/F
F/f/e
First(S) : [ a ].
First(B) : [ c ].
First(C) : [ b,e ].
First(E) : [ g,e ].
First(D) : [ g,e,f ].
First(F) : [ f,e ].
PS C:\college programs> █
```

PRACTICAL – 10

Write a program to check whether a grammar is operator
precedent.

Program:

```
#include<stdio.h>
#include<string.h>
char *input;
int i=0;
char lasthandle[6],stack[50],handles[][][5]={"")E","E*E","E+E","i","E^E"};
int top=0,l;
char prec[9][9]={
    '>','>','<','<','<','<','<','<','>','>',
    '>','>','<','<','<','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','>','e','e','>','>',
    '<','<','<','<','<','<','<','<','>','e',
    '>','>','>','>','>','e','e','>','>',
    '<','<','<','<','<','<','<','<','>','>',
    };
int getindex(char c)
{
switch(c)
{
case '+':return 0;
```

```

    case '-':return 1;
    case '*':return 2;
    case '/':return 3;
    case '^':return 4;
    case 'i':return 5;
    case '(':return 6;
    case ')':return 7;
    case '$':return 8;
}

int shift()
{
    stack[++top]=*(input+i++);
    stack[top+1]='\0';
}

int reduce()
{
    int i,len,found,t;
    for(i=0;i<5;i++)
    {
        len=strlen(handles[i]);
        if(stack[top]==handles[i][0]&&top+1>=len)
        {
            found=1;
            for(t=0;t<len;t++)
            {
                if(stack[top-t]!=handles[i][t])

```

```

    {
        found=0;
        break;
    }

    if(found==1)
    {
        stack[top-t+1]='E';
        top=top-t+1;
        strcpy(lasthandle,handles[i]);
        stack[top+1]='\0';
        return 1;
    }
}

return 0;
}

void dispstack()
{
    int j;
    for(j=0;j<=top;j++)
        printf("%c",stack[j]);
}

void dispinput()
{
    int j;
    for(j=i;j<l;j++)

```

```

    printf("%c",*(input+j));
}

void main()
{
int j;
input=(char*)malloc(50*sizeof(char));
printf("\nEnter the string\n");
scanf("%s",input);
input=strcat(input,"$");
l=strlen(input);
strcpy(stack,"$");
printf("\nSTACK\tINPUT\tACTION");
while(i<=l)
{
    shift();
    printf("\n");
    dispstack();
    printf("\t");
    dispinput();
    printf("\tShift");
    if(prec[getindex(stack[top])][getindex(input[i])]=='>
')
{
    while(reduce())
    {
        printf("\n");
        dispstack();
        printf("\t");
    }
}
}

```

```

        dispinput();
        printf("\tReduced: E->%s",lasthandle);
    }
}

if(strcmp(stack,"$E$")==0)
    printf("\nAccepted;");
else
    printf("\nNot Accepted;");
}

```

Output:

```

Enter the string
i*(i+i)*i

STACK   INPUT   ACTION
$i      *(i+i)*i$     Shift
$E      *(i+i)*i$     Reduced: E->i
$E*     (i+i)*i$     Shift
$E*(    i+i)*i$ Shift
$E*(i   +i)*i$ Shift
$E*(E   +i)*i$ Reduced: E->i
$E*(E+  i)*i$ Shift
$E*(E+i )*i$ Shift
$E*(E+E )*i$ Reduced: E->i
$E*(E   )*i$ Reduced: E->E+E
$E*(E)  *i$ Shift
$E*E   *i$ Reduced: E->)E(
$E   *i$ Reduced: E->E*E
$E*   i$ Shift
$E*i   $ Shift
$E*E   $ Reduced: E->i
$E   $ Reduced: E->E*E
$E$    Shift
$E$    Shift
Accepted;
PS C:\college programs> █

```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
P  BT  WT  TAT
1  10  0   10
2  5   10  15
3  8   15  23
Average waiting time = 8.333333
Average turn around time = 16.000000
PS C:\college programs>
```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Enter number of process: 5
Enter Burst Time:
P1: 01
P2: 02
P3: 03
P4: 04
P5: 05
P      BT      WT      TAT
P1     1       0       1
P2     2       1       3
P3     3       3       6
P4     4       6      10
P5     5      10      15
Average Waiting Time= 4.000000
Average Turnaround Time= 7.000000
PS C:\college programs> █
```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Enter Number of Processes: 5
Enter Burst Time and Priority Value for Process 1: 1 3
Enter Burst Time and Priority Value for Process 2: 2 2
Enter Burst Time and Priority Value for Process 3: 3 1
Enter Burst Time and Priority Value for Process 4: 4 5
Enter Burst Time and Priority Value for Process 5: 5 4
Order of process Execution is
P4 is executed from 0 to 4
P5 is executed from 4 to 9
P1 is executed from 9 to 10
P2 is executed from 10 to 12
P3 is executed from 12 to 15

Process Id    Burst Time   Wait Time   TurnAround Time
P4            4             0           4
P5            5             4           9
P1            1             9          10
P2            2            10          12
P3            3            12          15
PS C:\college programs> █
```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Total number of process in the system: 2

Enter the Arrival and Burst time of the Process[1]
Arrival time is: 1

Burst time is: 8

Enter the Arrival and Burst time of the Process[2]
Arrival time is: 2

Burst time is: 4
Enter the Time Quantum for the process: 3

Process No          Burst Time          TAT          Waiting Time
Process No[2]        4                  8           4
Process No[1]        8                  11          3
Average Turn Around Time: 9.500000
Average Waiting Time: 3.500000

```

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Enter no of pages:10
Enter the reference string:7 8 9 3 2 4 7 4 6 1
Enter no of frames:3

7
7     8
7     8     9
3     8     9
3     2     9
3     2     4
7     2     4
7     6     4
1     6     4

The no of page faults is 9
PS C:\college programs>

```

Incoming	t	Frame 1	t	Frame 2	t	Frame 3	
4		4		-		-	
1		4		1		-	
2		4		1		2	
4		4		1		2	
5		5		1		2	
Total Page Faults: 4							

```

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }

Enter number of frames: 3
Enter number of pages: 10
Enter page reference string: 7 8 9 3 2 1 6 5 4 9

7      -1      -1
7      8      -1
7      8      9
3      8      9
2      8      9
1      8      9
6      8      9
5      8      9
4      8      9
4      8      9

Total Page Faults = 9
PS C:\college programs>

```

```
Memory Management Scheme - Worst Fit
Enter the number of blocks:3
Enter the number of files:3

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4
File 3:5

File_no:      File_size :      Block_no:      Block_size:      Fragement
1            1              3                7                  6
2            4              1                5                  1
3            5              0                8                  0
```

```
Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File_no:      File_size :      Block_no:      Block_size:      Fragement
1            1              1                5                  4
2            4              3                7                  3
```

```
PS C:\college programs> █

Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:7
Block 3:2
Enter the size of the files :-
File 1:4
File 2:3

File No File Size      Block No      Block Size      Fragment
1          4             1              5                  1
2          3             2              7                  4
```

```
1 reader is inside  
1 Reader is leaving  
Writer is trying to enter  
Writer has entered  
Writer is leaving  
1 reader is inside  
1 Reader is leaving  
Writer is trying to enter  
Writer has entered  
Writer is leaving  
1 reader is inside  
1 Reader is leaving  
Writer is trying to enter  
Writer has entered  
Writer is leaving  
1 reader is inside  
1 Reader is leaving  
Writer is trying to enter  
Writer has entered  
Writer is leaving  
1 reader is inside  
1 Reader is leaving  
Writer is trying to enter  
Writer has entered  
Writer is leaving  
...Program finished with exit code 0  
Press ENTER to exit console.[]
```

```
Enter the number of Producers:2  
Enter the number of Consumers:2  
Enter buffer capacity:2  
Successfully created producer 1  
Producer 1 produced 29  
Successfully created producer 2  
Producer 2 produced 11  
Successfully created consumer 1  
Buffer:29 11  
Consumer 3 consumed 11  
Current buffer len: 1  
Buffer:29  
Consumer 2 consumed 29  
Current buffer len: 0  
Successfully created consumer 2  
Producer 1 produced 3  
Buffer:3  
Consumer 2 consumed 3  
Current buffer len: 0  
Producer 2 produced 31  
Producer 1 produced 27  
Producer 1 produced 7  
Buffer:31 27
```

```
Enter the number of process and resources  
5 3  
enter allocation of resource of all process 5x3 matrix  
1 2 3  
4 5 6  
0 6 7  
0 8 9  
9 5 4  
enter the max resource process required 5x3 matrix  
0 1 0  
1 2 3  
2 3 4  
3 4 5  
4 5 6  
enter the available resource 3 4 2  
  
need resources matrix are  
-1 -1 -3  
-3 -3 -3  
2 -3 -3  
3 -4 -4  
-5 0 2  
  
available resource after completion  
17 30 31  
safe sequence are  
p0 p1 p2 p3 p4
```

```
Enter the directory name:siddhika
Enter the number of files:3
Enter file name to be created:sid
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:shukla
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:ss
Do you want to enter another file(yes - 1 or no - 0):1
Enter file name to be created:sk
Do you want to enter another file(yes - 1 or no - 0):0
Directory name is:siddhika
Files names are:
sid
shukla
ss
sk
PS C:\college programs> █
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 1
```

```
Enter name of directory -- siddhika
Directory created
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 2
```

```
Enter name of the directory -- siddhika
Enter name of the file -- sid
File created
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 5
```

```
Directory      Files
siddhika      sid
```

```
1. Create Directory      2. Create File   3. Delete File
4. Search File         5. Display       6. Exit
Enter your choice -- 6
```

```
PS C:\college programs> █
```

```
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc hierarchical.c -o hierarchical } ; if (?) { .\hierarchical }
Enter number of users: 1
Enter name: siddhika
Enter dir(1) or file(0): sid
```

```
Hierarchical
█
```

GURU TEGH BAHADUR INSTITUTE OF
TECHNOLOGY

COMPILER DESIGN LAB FILE

NAME: SIDDHIKA SHUKLA

ENROLLMENT NO.: 05113203121

BRANCH: IT-1, 5TH SEM

TITLE TABLE

S.NO.	TITLE	PAGE NO.	SIGNATURE

PRACTICAL – 1

Introduction to Compiler Design.

The compiler is software that converts a program written in a high-level language (Source Language) to a low-level language (Object/Target/Machine Language/0, 1's).

A translator or language processor is a program that translates an input program written in a programming language into an equivalent program in another language. The compiler is a type of translator, which takes a program written in a high-level programming language as input and translates it into an equivalent program in low-level languages such as machine language or assembly language.

The program written in a high-level language is known as a source program, and the program converted into a low-level language is known as an object (or target) program. Without compilation, no program written in a high-level language can be executed. For every programming language, we have a different compiler; however, the basic tasks performed by every compiler are the same. The process of translating the source code into machine code involves several stages, including lexical analysis, syntax analysis, semantic analysis, code generation, and optimization.

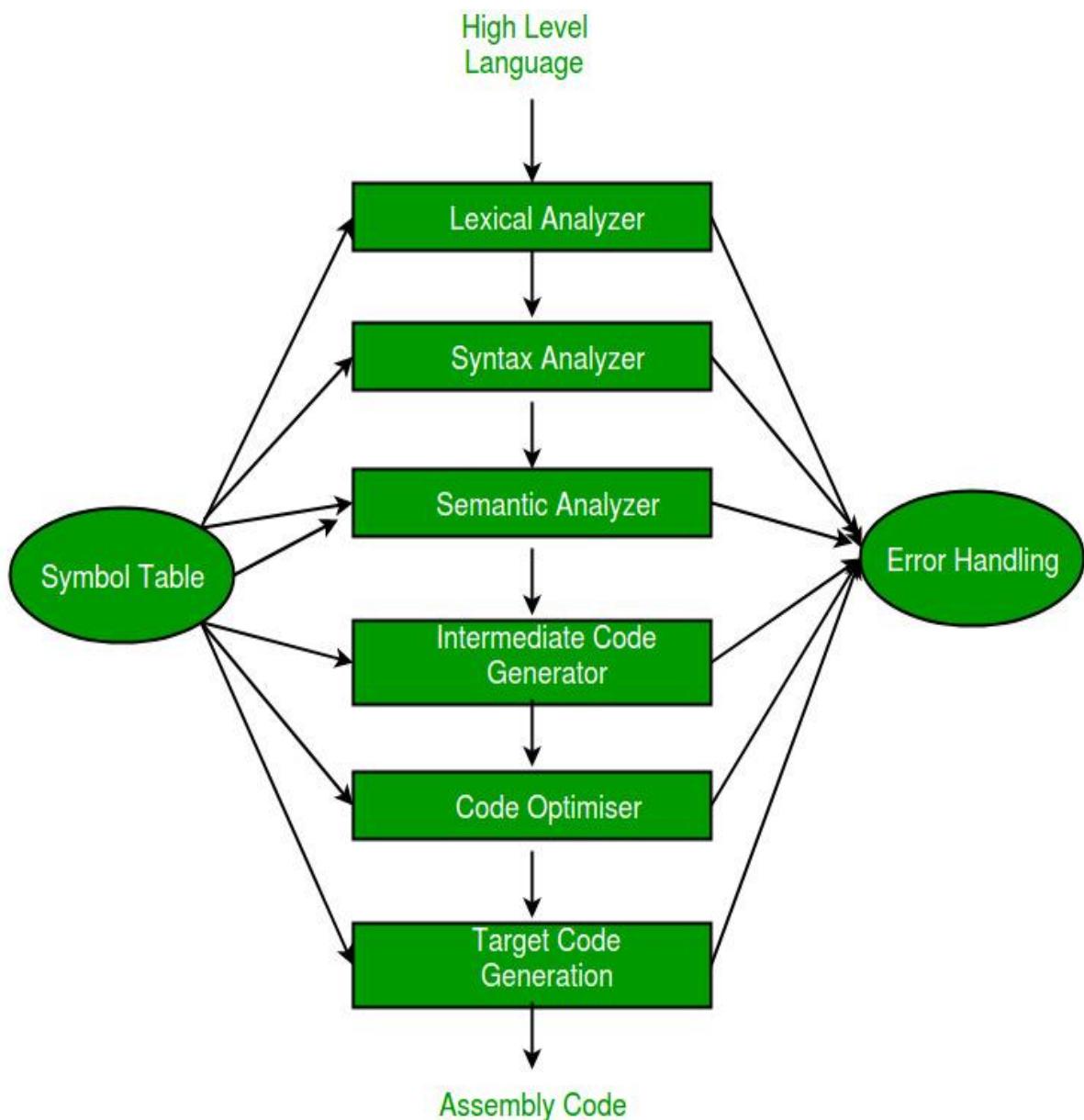
Compiler is an intelligent program as compare to an assembler. Compiler verifies all types of limits, ranges, errors, etc. Compiler program takes more time to run and it occupies huge amount of memory space. The speed of compiler is slower than other system software. It takes time because it enters through the program and then does translation of the full program. When compiler runs on same machine and produces machine code for the same machine on which it is running. Then it is called as self-compiler or resident compiler. Compiler may run on one machine and produces the machine codes for other computer then in that case it is called as cross compiler.

Stages of Compiler Design

- Lexical Analysis: The first stage of compiler design is lexical analysis, also known as scanning. In this stage, the compiler reads the source code character by character and breaks it down into a series of tokens, such as keywords, identifiers, and operators. These tokens are then passed on to the next stage of the compilation process.
- Syntax Analysis: The second stage of compiler design is syntax analysis, also known as parsing. In this stage, the compiler checks the syntax of the source code to ensure that it conforms to the rules of the programming language. The compiler builds a parse tree, which is a hierarchical representation of the program's structure, and uses it to check for syntax errors.
- Semantic Analysis: The third stage of compiler design is semantic analysis. In this stage, the compiler checks the meaning of the source code to ensure that it makes sense. The compiler performs type checking, which ensures that variables are used correctly and that operations are performed on compatible data types. The compiler also checks for other semantic errors, such as undeclared variables and incorrect function calls.
- Code Generation: The fourth stage of compiler design is code generation. In this stage, the compiler translates the parse tree into machine code that can be executed by the computer. The code generated by the compiler must be efficient and optimized for the target platform.
- Optimization: The final stage of compiler design is optimization. In this stage, the compiler analyses the generated code and makes optimizations to improve its performance. The compiler may perform optimizations such as constant folding, loop unrolling, and function inlining.

Overall, compiler design is a complex process that involves multiple stages and requires a deep understanding of both the programming language and the target platform. A well-designed compiler can

greatly improve the efficiency and performance of software programs, making them more useful and valuable for users.



PRACTICAL – 2

Write a program to check whether a string belong to the grammar or not.

Program:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main() {
    char string[50];
    int flag,count=0;
    printf("The grammar is:\n S->aS,\n S->Sb,\n S->ab\n");
    printf("Enter the string to be checked:\n");
    gets(string);
    if(string[0]=='a') {
        flag=0;
        for (count=1;string[count-1]!='\0';count++) {
            if(string[count]=='b') {
                flag=1;
                continue;
            }
            else if((flag==1)&&(string[count]=='a')) {
                printf("The string does not belong to the specified
grammar");
                break;
            }
            else if(string[count]=='a'){


```

```

        continue;

    }

else if((flag==1)&&(string[count]=='0')) {

    printf("String accepted.....!!!!");

    break;

}

else {

    printf("String not accepted");

}

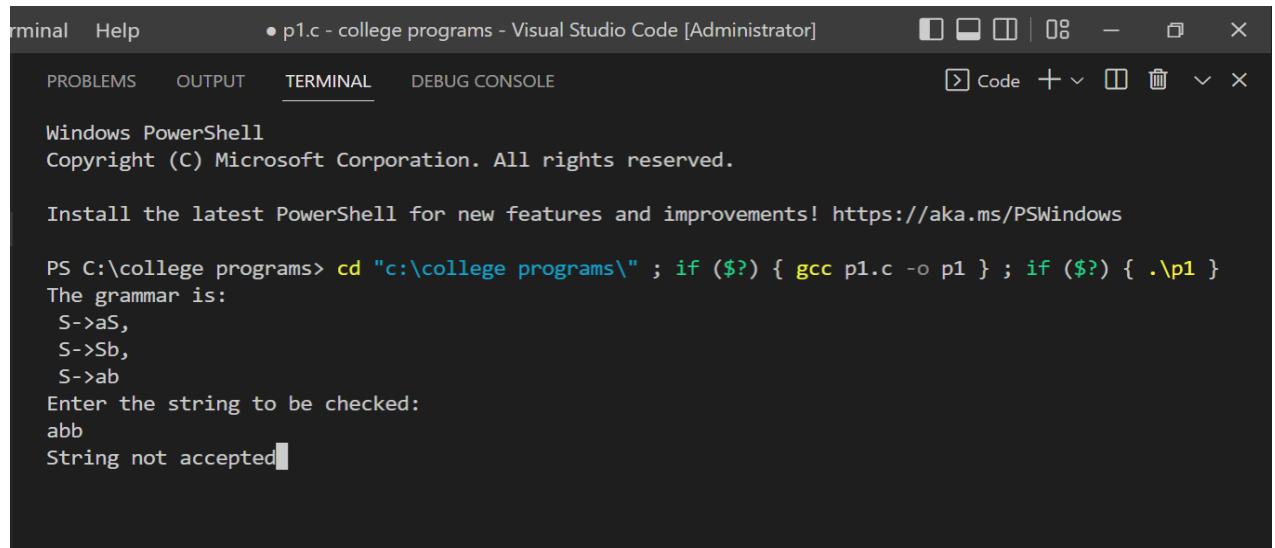
}

getch();

}

```

Output:



The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal tab is selected, showing the output of a C program. The program defines a grammar S -> aS, S -> Sb, and S -> ab. It then prompts the user to enter a string to be checked. The user enters 'abb', and the program outputs 'String not accepted'.

```

Terminal  Help      • p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  Code +  ×
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
The grammar is:
S->aS,
S->Sb,
S->ab
Enter the string to be checked:
abb
String not accepted

```

PRACTICAL – 3

Write a program to check whether a string include keyword or not.

Program:

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

bool isKeyword(char* str)
{
    if (!strcmp(str, "auto") || !strcmp(str, "default")
        || !strcmp(str, "signed") || !strcmp(str, "enum")
        || !strcmp(str, "extern") || !strcmp(str, "for")
        || !strcmp(str, "register") || !strcmp(str, "if")
        || !strcmp(str, "else") || !strcmp(str, "int")
        || !strcmp(str, "while") || !strcmp(str, "do")
        || !strcmp(str, "break") || !strcmp(str, "continue")
        || !strcmp(str, "double") || !strcmp(str, "float")
        || !strcmp(str, "return") || !strcmp(str, "char")
        || !strcmp(str, "case") || !strcmp(str, "const")
        || !strcmp(str, "sizeof") || !strcmp(str, "long")
        || !strcmp(str, "short") || !strcmp(str, "typedef")
        || !strcmp(str, "switch")
        || !strcmp(str, "unsigned") || !strcmp(str, "void")
        || !strcmp(str, "static") || !strcmp(str, "struct")
        || !strcmp(str, "goto") || !strcmp(str, "union")
        || !strcmp(str, "volatile")))
```

```

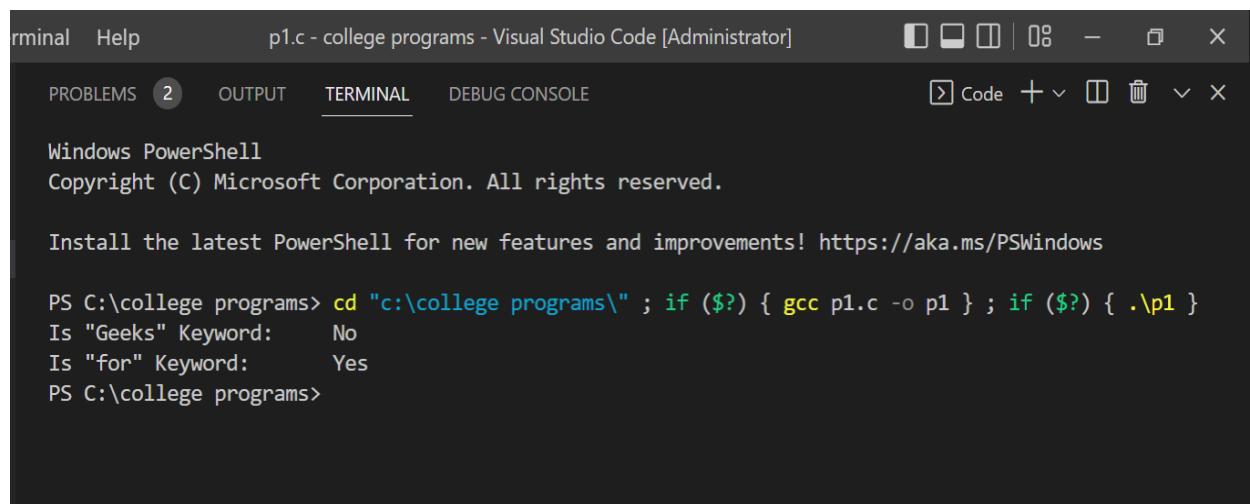
        return (true);

    return (false);
}

int main()
{
    printf("Is \"Geeks\" Keyword: \t");
    isKeyword("geeks") ? printf("Yes\n") : printf("No\n");
    printf("Is \"for\" Keyword: \t");
    isKeyword("for") ? printf("Yes\n") : printf("No\n");
    return 0;
}

```

Output:



The screenshot shows a Visual Studio Code interface with a terminal window open. The title bar reads "p1.c - college programs - Visual Studio Code [Administrator]". The terminal tab is selected, showing the following output:

```

Terminal  Help      p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS  2  OUTPUT  TERMINAL  DEBUG CONSOLE  Code  +  -  ×
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { ./p1 }
Is "Geeks" Keyword:      No
Is "for" Keyword:       Yes
PS C:\college programs>

```

PRACTICAL – 4

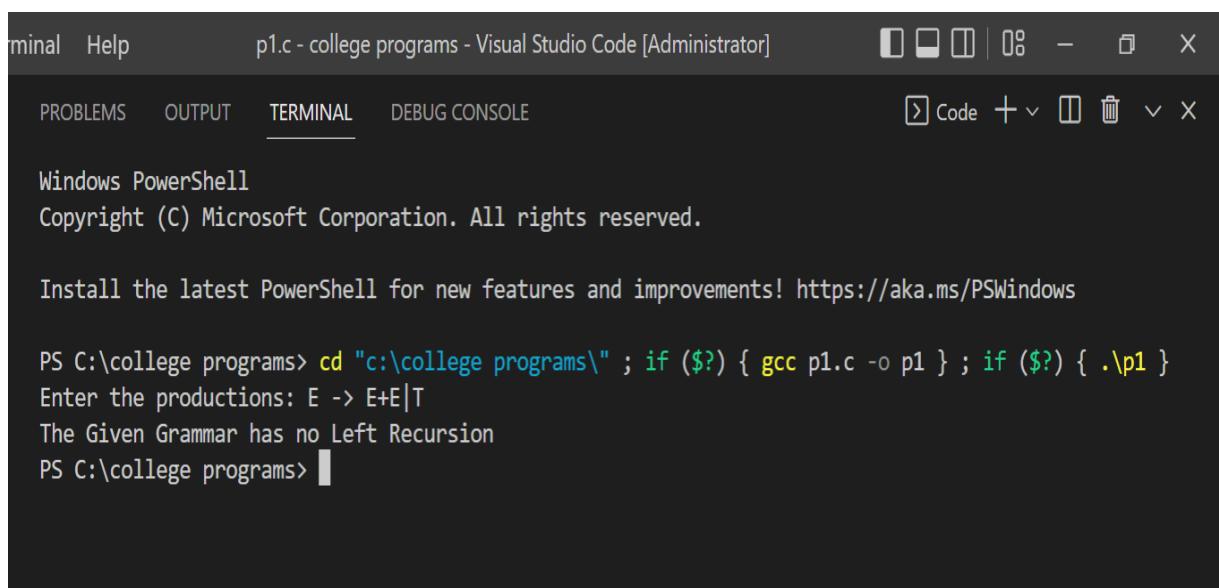
Write a program to remove left recursion from a grammar.

Program:

```
#include<stdio.h>
#include<string.h>
void main() {
    char input[100],l[50],r[50],temp[10],tempprod[20],productions[25][50];
    int i=0,j=0,flag=0,consumed=0;
    printf("Enter the productions: ");
    scanf("%1s->%s",l,r);
    printf("%s",r);
    while(sscanf(r+consumed,"%[^]s",temp) == 1 && consumed <= strlen(r)) {
        if(temp[0] == l[0]) {
            flag = 1;
            sprintf(productions[i++],"%s->%s%0s'\0",l,temp+1,l);
        }
        else
            sprintf(productions[i++],"%s'->%s%0s'\0",l,temp,l);
        consumed += strlen(temp)+1;
    }
    if(flag == 1) {
        sprintf(productions[i++],"%s->\0",l);
        printf("The productions after eliminating Left Recursion are:\n");
        for(j=0;j<i;j++)
            printf("%s\n",productions[j]);
    }
}
```

```
else
    printf("The Given Grammar has no Left Recursion");
}
```

Output:



A screenshot of the Visual Studio Code interface, specifically the terminal tab. The title bar shows "p1.c - college programs - Visual Studio Code [Administrator]". The terminal window displays the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
Enter the productions: E -> E+E|T
The Given Grammar has no Left Recursion
PS C:\college programs>
```

PRACTICAL – 5

Write a program to perform left factoring on a grammar.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
    char
gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[20];
    int i,j=0,k=0,l=0,pos;
    printf("Enter Production : A->");
    gets(gram);
    for(i=0;gram[i]!='|';i++,j++)
        part1[j]=gram[i];
    part1[j]='\0';
    for(j=++i,i=0;gram[j]!='\0';j++,i++)
        part2[i]=gram[j];
    part2[i]='\0';
    for(i=0;i<strlen(part1)||i<strlen(part2);i++){
        if(part1[i]==part2[i]){
            modifiedGram[k]=part1[i];
            k++;
            pos=i+1;
        }
    }
    for(i=pos,j=0;part1[i]!='\0';i++,j++){

```

```

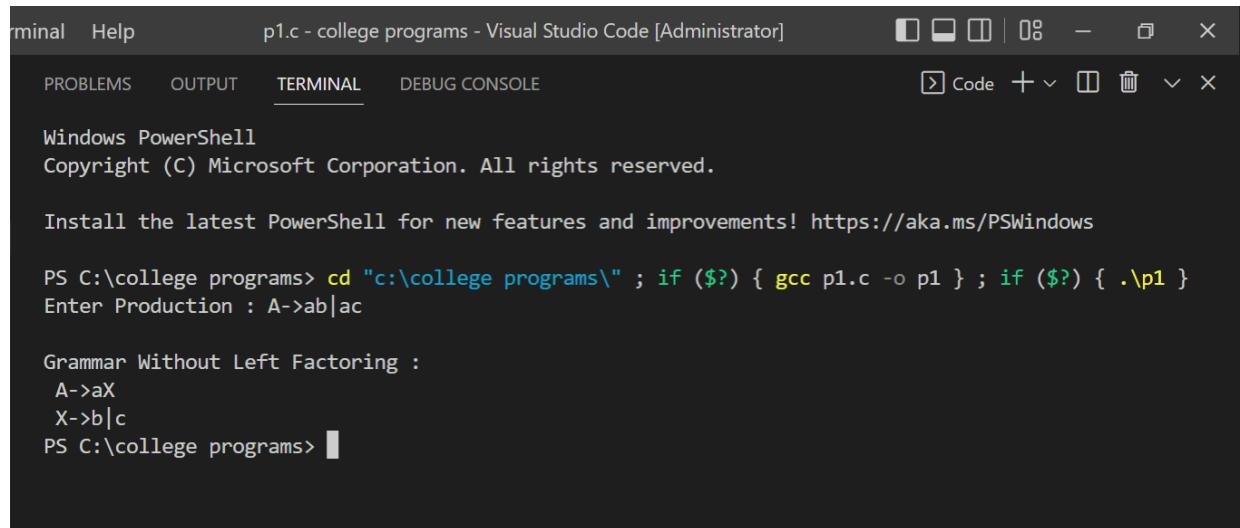
    newGram[j]=part1[i];
}

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

modifiedGram[k]='X';
modifiedGram[++k]='\0';
newGram[j]='\0';
printf("\nGrammar Without Left Factoring : \n");
printf(" A->%s",modifiedGram);
printf("\n X->%s\n",newGram);
}

```

Output:



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal tab is selected, showing the Windows PowerShell environment. The output displays the original grammar rules, the transformed grammar without left factoring, and the resulting modified and new grammars.

```

Terminal Help p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Code + ×

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
Enter Production : A->ab|ac

Grammar Without Left Factoring :
A->aX
X->b|c
PS C:\college programs>

```

PRACTICAL – 6

Write a program to show all the operations of a stack.

Program:

```
#include <stdio.h>
int MAXSIZE = 8;
int stack[8];
int top = -1;
int isempty(){
    if(top == -1)
        return 1;
    else
        return 0;
}
int isfull(){
    if(top == MAXSIZE)
        return 1;
    else
        return 0;
}
int peek(){
    return stack[top];
}
int pop(){
    int data;
    if(!isempty()) {
        data = stack[top];
        top--;
    }
    return data;
}
```

```

    top = top - 1;
    return data;
} else {
    printf("Could not retrieve data, Stack is empty.\n");
}
}

int push(int data){
    if(!isfull()) {
        top = top + 1;
        stack[top] = data;
    } else {
        printf("Could not insert data, Stack is full.\n");
    }
}

int main(){
    push(44);
    push(10);
    push(62);
    push(123);
    push(15);
    printf("Element at top of the stack: %d\n",peek());
    printf("Elements: \n");

    // print stack data
    while(!isempty()) {
        int data = pop();
        printf("%d\n",data);
    }
}

```

```

    }

printf("Stack full: %s\n" , isfull()?"true":"false");

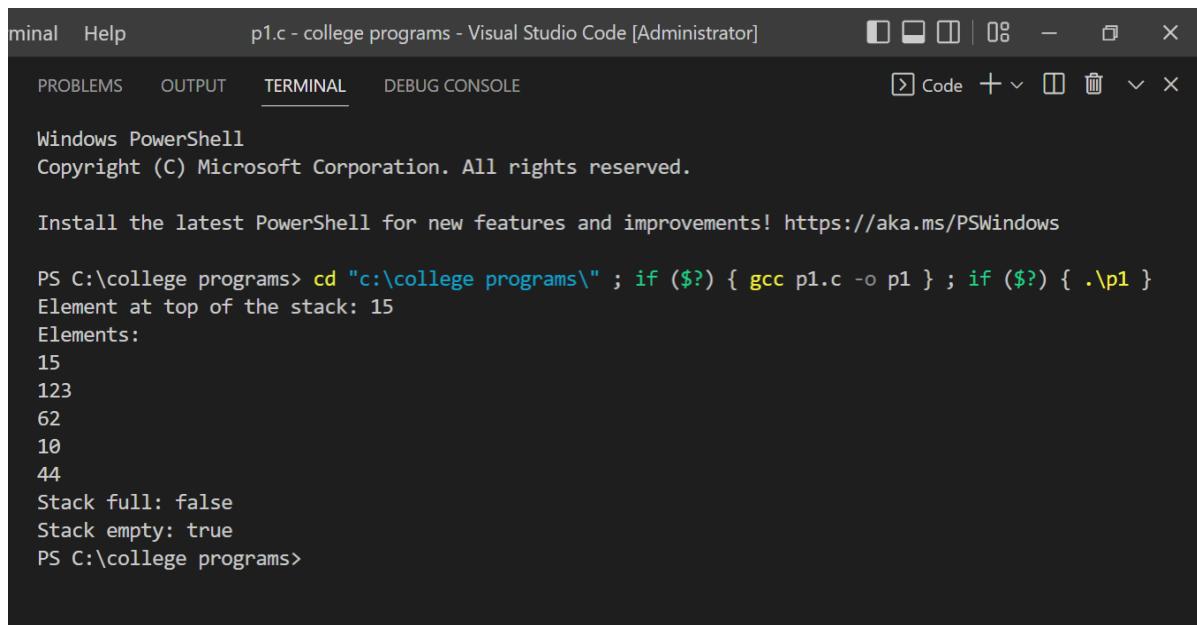
printf("Stack empty: %s\n" , isempty()?"true":"false");

return 0;

}

```

Output:



The screenshot shows a Windows PowerShell terminal window within the Visual Studio Code interface. The title bar indicates the file is p1.c - college programs - Visual Studio Code [Administrator]. The terminal tab is active, showing the following command-line session:

```

minal Help          p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
Element at top of the stack: 15
Elements:
15
123
62
10
44
Stack full: false
Stack empty: true
PS C:\college programs>

```

PRACTICAL – 7

Write a program to find out the leading of the non-terminals in a grammar.

Program:

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

char arr[18][3]={{'E', '+', 'F'}, {'E', '*', 'F'}, {'E', '(', 'F'}, {'E', ')', 'F'}, {'E', 'i', 'F'}, {'E', '$', 'F'},
{'F', '+', 'F'}, {'F', '*', 'F'}, {'F', '(', 'F'}, {'F', ')', 'F'}, {'F', 'i', 'F'}, {'F', '$', 'F'}, {'T', '+', 'F'},
{'T', '*', 'F'}, {'T', '(', 'F'}, {'T', ')', 'F'}, {'T', 'i', 'F'}, {'T', '$', 'F'}};
char prod[6] = "EETTFF";
char res[6][3]={{'E', '+', 'T'}, {'T', '\0'}, {'T', '*', 'F'}, {'F', '\0'}, {'(' , 'E', ')'}, {'i', '\0'}};
char stack [5][2];
int top = -1;

void install(char pro, char re) {
    int i;
    for (i = 0; i < 18; ++i) {
        if (arr[i][0] == pro && arr[i][1] == re) {
            arr[i][2] = 'T';
            break;
        }
    }
    ++top;
    stack[top][0] = pro;
```

```

stack[top][1] = re;
}

void main() {
    int i = 0, j;
    char pro, re, pri = ' ';
    for (i = 0; i < 6; ++i) {
        for (j = 0; j < 3 && res[i][j] != '\0'; ++j) {
            if (res[i][j] == '+' || res[i][j] == '*' || res[i][j] == '(' || res[i][j] == ')' ||
                res[i][j] == 'i' || res[i][j] == '$') {
                install(prod[i], res[i][j]);
                break;
            }
        }
    }
    while (top >= 0) {
        pro = stack[top][0];
        re = stack[top][1];
        --top;
        for (i = 0; i < 6; ++i) {
            if (res[i][0] == pro && res[i][0] != prod[i]) {
                install(prod[i], re);
            }
        }
    }
    for (i = 0; i < 18; ++i) {
        printf("\n\t");
        for (j = 0; j < 3; ++j)
            printf("%c\t", arr[i][j]);
    }
}

```

```

}

getch();

printf("\n\n");

for (i = 0; i < 18; ++i) {

    if (pri != arr[i][0]) {

        pri = arr[i][0];

        printf("\n\t%c -> ", pri);

    }

    if (arr[i][2] == 'T')

        printf("%c ", arr[i][1]);

    }

getch();

}

```

Output:

```

Terminal  Help          p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  Code  +  □  ▾  ×
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }

      E      +      T
      E      *      T
      E      (      T
      E      )      F
      E      i      T
      E      $      F
      F      +      F
      F      *      F
      F      (      T
      F      )      F
      F      i      T
      F      $      F
      T      +      F
      T      *      T
      T      (      T
      T      )      F
      T      i      T
      T      $      F

      E -> + * ( i
      F -> ( i
      T -> * ( i
PS C:\college programs>

```

PRACTICAL – 8

Write a program to implement shift reduce parsing for a string.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
char ip_sym[15],stack[15];
int ip_ptr=0,st_ptr=0,len,i;
char temp[2],temp2[2];
char act[15];
void check();
void main()
{
    printf("\n\t\t SHIFT REDUCE PARSER\n");
    printf("\n GRAMMER\n");
    printf("\n E->E+E\n E->E/E");
    printf("\n E->E*E\n E->a/b");
    printf("\n enter the input symbol: ");
    gets(ip_sym);
    printf("\n\t stack implementation table");
    printf("\n stack \t\t input symbol\t\t action");
    printf("\n _____ \t\t _____ \t\t _____\n");
    printf("\n \$\t\t%$%\t\t-",ip_sym);
    strcpy(act,"shift ");
```

```

temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
len=strlen(ip_sym);
for(i=0;i<=len-1;i++)
{
stack[st_ptr]=ip_sym[ip_ptr];
stack[st_ptr+1]='\0';
ip_sym[ip_ptr]=' ';
ip_ptr++;
printf("\n $%s\t\t%s$\t\t%s",stack,ip_sym,act);
strcpy(act,"shift ");
temp[0]=ip_sym[ip_ptr];
temp[1]='\0';
strcat(act,temp);
check();
st_ptr++;
}
st_ptr++;
check();
}

void check()
{
int flag=0;
temp2[0]=stack[st_ptr];
temp2[1]='\0';
if((!strcmpi(temp2,"a"))||(!strcmpi(temp2,"b")))

```

```

{
stack[st_ptr]='E';
if(!strcmpi(temp2,"a"))
printf("\n $%s\t\t%s$\t\tE->a",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\tE->b",stack,ip_sym);
flag=1;
}
if((!strcmpi(temp2,"+"))||(!strcmpi(temp2,"*"))||(!strcmpi(temp2,"/"))) { flag=1;
}
if((!strcmpi(stack,"E+E"))||(!strcmpi(stack,"E\E"))||(!strcmpi(stack,"E*E")))
{
strcpy(stack,"E");
st_ptr=0;
if(!strcmpi(stack,"E+E"))
printf("\n $%s\t\t%s$\t\tE->E+E",stack,ip_sym);
else
if(!strcmpi(stack,"E\E"))
printf("\n $%s\t\t%s$\t\tE->E\E",stack,ip_sym);
else if(!strcmpi(stack,"E*E"))
printf("\n $%s\t\t%s$\t\tE->E*E",stack,ip_sym);
else
printf("\n $%s\t\t%s$\t\tE->E+E",stack,ip_sym);
flag=1;
}
if(!strcmpi(stack,"E")&&ip_ptr==len)
{

```

```

printf("\n $%s\t\t%s$\t\tACCEPT",stack,ip_sym);

getch();

exit(0);

}

if(flag==0)

{

printf("\n%s\t\t%s$\t\t reject",stack,ip_sym);

exit(0);

}

return;
}

```

Output:

The screenshot shows a Visual Studio Code interface with a terminal window open. The terminal title is "p1.c - college programs - Visual Studio Code [Administrator]". The terminal content displays the following output:

```

terminal Help p1.c - college programs - Visual Studio Code [Administrator]
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\college programs> cd "c:\college programs\" ; if ($?) { gcc p1.c -o p1 } ; if (?) { .\p1 }
SHIFT REDUCE PARSER

GRAMMER

E->E+E
E->E/E
E->E*E
E->a/b
enter the input symbol: a+b

stack implementation table
-----      -----
stack      input symbol      action
-----      -----
$          a+b$              --
$a         +b$              shift a
$E         +b$              E->a
$E+
$E+b      b$              shift +
$E+E      $                shift b
$E         $                E->b
$E         $                E->E+E
$E         $                ACCEPT
PS C:\college programs>

```

PRACTICAL – 9

Write a program to find out the FIRST of the non-terminals in a grammar.

Program:

```
#include<stdio.h>
#include<conio.h>
char array[10][20],temp[10];
int c,n;
void fun(int,int[]);
int fun2(int i,int j,int p[],int );
void main()
{
    int p[2],i,j;
    printf("Enter the no. of productions :");
    scanf("%d",&n);
    printf("Enter the productions :\n");
    for(i=0;i<n;i++)
        scanf("%s",array[i]);
    for(i=0;i<n;i++)
    {
        c=-1,p[0]=-1,p[1]=-1;
        fun(i,p);
        printf("First(%c) : [ ",array[i][0]);
        for(j=0;j<=c;j++)
            printf("%c,",temp[j]);
        printf("\b ].\n");
    }
}
```

```

getch();
}

}

int fun2(int i,int j,int p[],int key)

{
    int k;
    if(!key)
    {
        for(k=0;k<n;k++)
        if(array[i][j]==array[k][0])
        break;
        p[0]=i;p[1]=j+1;
        fun(k,p);
        return 0;
    }
    else
    {
        for(k=0;k<=c;k++)
        {
            if(array[i][j]==temp[k])
            break;
        }
        if(k>c)return 1;
        else return 0;
    }
}
void fun(int i,int p[])

```

```

{
int j,k,key;
for(j=2;array[i][j] != NULL; j++)
{
if(array[i][j-1]=='/')
{
if(array[i][j]>= 'A' && array[i][j]<='Z')
{
key=0;
fun2(i,j,p,key);
}
else
{
key = 1;
if(fun2(i,j,p,key))
temp[++c] = array[i][j];
if(array[i][j]== '@'&& p[0]!=-1)
{
if(array[p[0]][p[1]]>='A' && array[p[0]][p[1]] <='Z')
{
key=0;
fun2(p[0],p[1],p,key);
}
else
if(array[p[0]][p[1]] != '/'&& array[p[0]][p[1]]!=NULL)
{
if(fun2(p[0],p[1],p,key))

```

```
temp[++c]=array[p[0]][p[1]];
}
}
}
}
}
}
```

Output:

```
Enter the no. of productions :6
Enter the productions :
S/aBDh
B/cC
C/bC/e
E/g/e
D/E/F
F/f/e
First(S) : [ a ].
First(B) : [ c ].
First(C) : [ b,e ].
First(E) : [ g,e ].
First(D) : [ g,e,f ].
First(F) : [ f,e ].
PS C:\college programs> █
```

PRACTICAL – 10

Write a program to check whether a grammar is operator
precedent.

Program:

```
#include<stdio.h>
#include<string.h>
char *input;
int i=0;
char lasthandle[6],stack[50],handles[][][5]={"")E","E*E","E+E","i","E^E"};
int top=0,l;
char prec[9][9]={
    '>','>','<','<','<','<','<','<','>','>',
    '>','>','<','<','<','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','<','<','<','>','>',
    '>','>','>','>','>','e','e','>','>',
    '<','<','<','<','<','<','<','<','>','e',
    '>','>','>','>','>','e','e','>','>',
    '<','<','<','<','<','<','<','<','>','>',
    };
int getindex(char c)
{
switch(c)
{
case '+':return 0;
```

```

        case '-':return 1;
        case '*':return 2;
        case '/':return 3;
        case '^':return 4;
        case 'i':return 5;
        case '(':return 6;
        case ')':return 7;
        case '$':return 8;
    }
}

int shift()
{
    stack[++top]=*(input+i++);
    stack[top+1]='\0';
}

int reduce()
{
    int i,len,found,t;
    for(i=0;i<5;i++)
    {
        len=strlen(handles[i]);
        if(stack[top]==handles[i][0]&&top+1>=len)
        {
            found=1;
            for(t=0;t<len;t++)
            {
                if(stack[top-t]!=handles[i][t])

```

```

    {
        found=0;
        break;
    }

    if(found==1)
    {
        stack[top-t+1]='E';
        top=top-t+1;
        strcpy(lasthandle,handles[i]);
        stack[top+1]='\0';
        return 1;
    }
}

return 0;
}

void dispstack()
{
    int j;
    for(j=0;j<=top;j++)
        printf("%c",stack[j]);
}

void dispinput()
{
    int j;
    for(j=i;j<l;j++)

```

```

    printf("%c",*(input+j));
}

void main()
{
int j;
input=(char*)malloc(50*sizeof(char));
printf("\nEnter the string\n");
scanf("%s",input);
input=strcat(input,"$");
l=strlen(input);
strcpy(stack,"$");
printf("\nSTACK\tINPUT\tACTION");
while(i<=l)
{
    shift();
    printf("\n");
    dispstack();
    printf("\t");
    dispinput();
    printf("\tShift");
    if(prec[getindex(stack[top])][getindex(input[i])]=='>
')
{
    while(reduce())
    {
        printf("\n");
        dispstack();
        printf("\t");
    }
}
}

```

```

        dispinput();
        printf("\tReduced: E->%s",lasthandle);
    }
}

if(strcmp(stack,"$E$")==0)
    printf("\nAccepted;");
else
    printf("\nNot Accepted;");
}

```

Output:

```

Enter the string
i*(i+i)*i

STACK   INPUT   ACTION
$i      *(i+i)*i$     Shift
$E      *(i+i)*i$     Reduced: E->i
$E*     (i+i)*i$     Shift
$E*(    i+i)*i$ Shift
$E*(i   +i)*i$ Shift
$E*(E   +i)*i$ Reduced: E->i
$E*(E+  i)*i$ Shift
$E*(E+i )*i$ Shift
$E*(E+E )*i$ Reduced: E->i
$E*(E   )*i$ Reduced: E->E+E
$E*(E)  *i$ Shift
$E*E   *i$ Reduced: E->)E(
$E   *i$ Reduced: E->E*E
$E*   i$ Shift
$E*i   $ Shift
$E*E   $ Reduced: E->i
$E   $ Reduced: E->E*E
$E$    Shift
$E$    Shift
Accepted;
PS C:\college programs> █

```

EASE

OCTOBER
2023

S	M	T	W	T	F	S
1 BEDTIME YOGA SEQUENCE 37 min	2 SUKHASANA, THE EASY POSE 8 min MORNING YOGA FOR BEGINNERS - GENTLE MORNING YOGA 21 min	3 NEW YOGA FOR SICK RECOVERY 21 min	4 YOGA CAMP - DAY 11 I RELEASE 45 min	5 FUNDAMENTALS OF EASE 35 min	6 YOGA FOR STRENGTH AND FOCUS 43 min	7 YOGA FOR RELAXATION 33 min FWFG: Ease of Fight or Flight 27 min
8 CROW POSE 10 min CROW PRACTICE 24 min FWFG: Ease in the Seat of Fire - 25 min	9 QUICK STRESS FIX 5 min MEDITATION FOR INNER PEACE 11 min	10 STRESS MELT 25 min	11 YOGA TO CALM YOUR NERVES 24 min	12 30 DAYS OF YOGA - DAY 13 - ENDURANCE & EASE 26 min	13 STREET YOGA - YOGA YOU CAN DO ANYWHERE 14 min	14 PEACEFUL WARRIOR 27 min
15 INTRO TO YIN YOGA 26 min FWFG: Functional Movement with Sumair - Day 1 - 24 min	16 30 DAYS OF YOGA - DAY 1 - EASE INTO IT 35 min FWFG: Functional Movement with Sumair - Day 2 - 37 min	17 YOGA FOR PANIC & ANXIETY 15 min FWFG: Functional Movement with Sumair - Day 3 - 20 min	18 MORNING YOGA FLOW 20 min FWFG: Functional Movement with Sumair - Day 4 - 11 min	19 YOGA TO SLOW YOUR ROLL 16 min FWFG: Functional Movement with Sumair - Day 5 - 13 min	20 YOGA FOR STRESS RELIEF 37 min FWFG: Functional Movement with Sumair Day 6 - 14 min	21 REVOLUTION - DAY 1 - PRACTICE EASE 24 min FWFG: Functional Movement with Sumair - Day 7 - 13 min
22 YOGA FOR WHEN YOU ARE SICK 20 min	23 FOUNDATIONS OF YOGA - WILD THING 3 min YOGA FOR COMFORT AND NOURISHMENT 25 min	24 MEDITATION FOR BALANCING THE NERVOUS SYSTEM 11 min	25 YOGA FOR ZOMBIES 15 min	26 YOGA FOR HEALTHY BLOOD FLOW 19 min	27 YOGA FOR INNER SPACE TRAVEL 14 min	28 YOGA FOR WHEN YOU FEEL DEAD INSIDE 27 min
29 YOGA FOR WHEN YOU ARE FEELING SCARED 28 min	30 YOGA FOR BONE HEALTH 21 min	31 NEW BRIDE OF PLANKENSPINE PRACTICE 18 min	Free practices all month long on YWA YouTube! Check out FindWhatFeelsGood.com for Adriene's monthly member's vlog + other new uploads! Join us 10/15-10/21 for a brand new 7 Day Functional Movement with Sumair on FWFG.			



OCTOBER
2023

EASE

Oh sweet soft supportive October. Welcome. A new month.
And with it, another free yoga practice calendar to help you slink into daily practice and feel your sacred center.

Ease is what this month is about.

Strength with ease. Curiosity and ease. Creativity and ease. Rest with ease.

If there was ever a month to give the full 31 practices a try, this may be it. With a calming yet strengthening and fun curation, this set of sessions is sweet, spooky and designed to inspire the spontaneity of ease.

Allow ease in, let it surprise you.

Softens, let go.

I will see you on the mat!

October disclosure 🎃: It is a tradition that we do a Halloween video for all ages on the Yoga With Adriene channel - so you will see a week of Halloween themed practices including a brand new one on Halloween October 31. These are fun practices to inspire a playfulness and ease. Practice alone if you dare, or invite a friend to join you. (No tricks, just treats.)

FWFG members, join me for Sumair's Functional Movement Series this month! I challenge you to take the whole week with me! No stops. You won't regret it. Working with a professional trainer is an amazing way to deepen your practice and knowledge of your body to increase ability, ease, and avoid injury.

*Daily practices are free on YouTube

*Find What Feels Good (FWFG) practices may be found at FWFG.com, thank you for being a member!

Much Love,
Adriene



Playlists

- Bookmark the playlist for easy access.
 - free YouTube playlist
 - FWFG playlist for members
- Learn more about the FWFG videos on the calendar by visiting FWFG.com.

How it Works

- Follow along every day or drop in throughout the month.
- Invite someone to join you!
- Share your experience with the community. Follow the hashtag and tag your social posts with it.
- Connect with @adrienelouise and @fwfglife on Instagram.

#ywaEASE



FREE PROGRAMMING RESOURCES FROM THE LEARN PROGRAMMING ACADEMY



LearnProgrammingAcademy.com





ABOUT THE LEARN PROGRAMMING ACADEMY (LPA)

The Learn Programming Academy produces world-class video training to teach people how to become programmers in the shortest possible time, utilizing industry best practices to ensure the training is 100% relevant to today's employers.

We're on a mission to teach one million people how to become programmers and are already over fifty percent of the way!

[BEGINNING C++ COURSE SLIDES \(FREE\)](#)

[FREE YOUTUBE LEARN TO CODE COURSE](#)

[OUR 53 PROGRAMMING COURSES](#)

[PROGRAMMING CAREER GUIDE E-BOOK](#)

[PROGRAMMING CAREER VIDEOS](#)

Learn Programming Academy Headquarters
Level 1, 8 Beulah Road, Norwood SA 5067
Australia

V1.1 - © Copyright 2021
Learn Programming Academy Pty Ltd
May be shared with others if copyright and credit left intact.

LearnProgrammingAcademy.com

Table of Contents

- Chapter 1 -** [Introduction](#)
- Chapter 2 -** [Beginning C++ Course Slides](#)
- Chapter 3 -** [Programming Career Guide](#)
- Chapter 4 -** [List of Learn Programming Academy Courses](#)
- Chapter 5 -** [Learn to Code Course](#)
- Chapter 6 -** [Programming Tip of the Day videos](#)

CHAPTER 1 – INTRODUCTION

Introduction - Thanks for downloading this E-Book.

Subscribe to my Email list ** groan **.

To get a lot of the free stuff, I'm asking you to sign up for my email list. Why am I asking you to sign up to my email list?

1. Getting course updates to you!

Firstly, I want to keep you updated when new courses come out. The reality is the announcements on Udemy seem only ever to be received by around 2% or less of my students. The remaining 98% don't see these announcements.

Why? I don't know. I know that about 98% of my students never see some or all of the Udemy announcements I send. **Getting you to sign up for my email list makes it easy for me to keep you informed.**

Most of my communication moving forward will be regular updates on new and updated courses I have published, free stuff like the new career paths guide, my existing programming career guide, my new Learn to Code course, my 100+ programming tips of the day videos, and more.

Plus, as a student of Beginning C++, you also get the current version of the slides for the course and the updates when they are released.

2. Making my courses as cheap as possible for you.

In early October 2019, Udemy changed the way that instructors can provide discounts for courses.

It's no longer possible for an instructor to create a never expiring coupon for their courses at the lowest price.

I, along with other instructors on Udemy, can only pass on the biggest discounts to our courses with coupons that expire automatically in five days from the date of creation.

I'll also send out a Udemy promotion, but as established above, around 98% of students on Udemy never see those announcements.

Because of the limited number of coupons that I can create each month, I'll be sending out emails to students on my list with those discounts once or twice a month. This way, I can ensure that I pass on the biggest discounts.

These links I will send will enable you to buy any of my courses at the lowest price.

Make sure you sign up and say on my email list if you want to get access to these discounts.

3. So I can spam you?

No, I won't.

I promise not to send too many emails, and I won't spam. The only information you will get relates to my Learn Programming academy's courses or free programming stuff that is useful and will be of benefit to you on your programming journey.

While I will be upfront and say I hope you will consider purchasing some of my other programming courses if they prove useful and fit a need you have to learn a specific programming language or framework, I am not about 'selling.'

You won't ever see me sending you "stuff" to buy (other than my courses).

I'm about empowering people to become programmers by giving them the training and resources they need to succeed.

CHAPTER 2 – BEGINNING C++ SLIDES

Here is where you get access to the slides for the Beginning C++ course.

Note: I am doing it this way so that I can send out an email every time the slides are updated.

Note that by clicking the link below, you also get the programming career book.

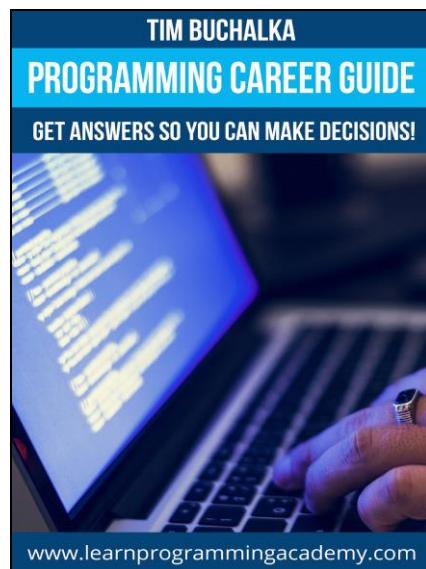
[Click HERE to get the Beginning C++ Slides and the programming career E-Book.](#)

CHAPTER 3 – PROGRAMMING CAREER GUIDE

This Programming Career guide answers a lot of the questions that students have relating to careers in programming. Having trained over half a million students to program, I've identified the major questions and wrote an E-Book to answer them.

To grab this Guide right now, 100% for free, click the link below.

[CLICK HERE TO GET BOTH PROGRAMMING CAREER GUIDES \(PLUS AS A STUDENT OF THE BEGINNING C++ COURSE YOU ALSO GET THE SLIDES FOR THE COURSE\).](#)



OTHER STUFF

Here are some other very useful resources for you - these don't require you to subscribe to an email list, click the link below to find out more about each course, or to watch the Learn to Code Video Series, or to watch 106 Programming Tip of the day videos I have created.

No strings attached!

CHAPTER 4 – COMPLETE LIST OF OUR COURSES

Currently, the Academy has fifty (52) software development courses. Languages such as Java, Python, C++, C#, C, Ruby, Kotlin, and more are covered and frameworks such as Android app development, Spring Framework, Java Enterprise development, Games development.

All these courses are on Udemy.

Below is a screenshot of each course, and you can click the image or link below for more information about each course.

Note: The links will give you a great discount price off the “everyday” price of each course on Udemy.

But keep in mind that due to Udemy changing their system for coupons, I can no longer create coupons that do not expire.

If you want the biggest possible discounted price for each course, then [join my email list](#) – once or twice a month, I'll send through coupons that offer all my courses at the lowest cost. Keep in mind these coupons will expire in five days from the email I send.

As an example, as of the time of writing this e-Book, if you click a link to one of my courses below, then the price should be **USD 13.99** (the minimum price Udemy allows me to set for a coupon that expires in 31 days).

If you sign up to my email list and wait for the next email I sent out (generally once or twice a month), then the price is **USD 9.99** – as you can see a considerable discount. In this case, the link will expire in only five days, hence the need for you to be signed up to receive the emails.

Keep in mind that Udemy automatically changes pricing into your local currency, but the percentage of discounts should be the same.

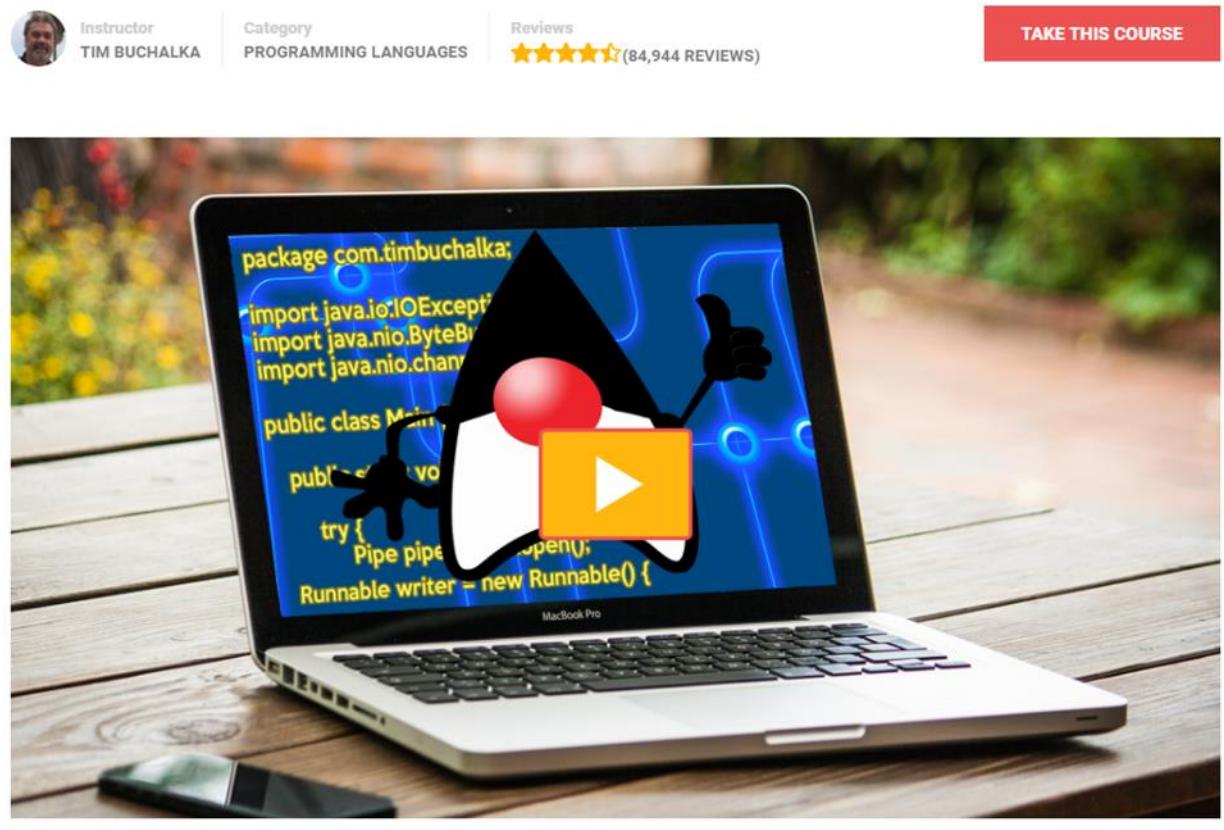
Either way, you get a considerable discount from the every day Udemy price for one of my courses.

You have two options. Get the course today for a still decent discount or join the email list for a much larger discount and then wait until I send out the discount links each month.

I hope that makes sense – send me a message if anything is unclear.

Java Programming Masterclass for Software Developers

Learn Java In This Course And Become a Computer Programmer. Obtain valuable Core Java Skills And Java Certification

A screenshot of a course page for "Java Programming Masterclass for Software Developers". At the top, there's a navigation bar with "FREE Programming Resources" and the "Learn Programming .academy" logo. Below the navigation, there's a large image of a laptop displaying Java code on its screen. The code includes imports for java.io.IOException, java.nio.ByteBuffer, and java.nio.channels, followed by a Main class definition. A video player interface with a play button is overlaid on the screen. The laptop is placed on a wooden surface outdoors, with a smartphone visible next to it. The course page includes sections for "Instructor" (Tim Buchalka), "Category" (PROGRAMMING LANGUAGES), "Reviews" (4.5 stars, 84,944 reviews), and a prominent red "TAKE THIS COURSE" button.

Get the [Java Masterclass here.](#)

Learn Python Programming Masterclass

This Python For Beginners Course Teaches You The Python Language Fast. Includes Python Online Training With Python 3



Instructor
TIM BUCHALKA

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (28,174 REVIEWS)

TAKE THIS COURSE



Get the [Python Masterclass here.](#)

Beginning C++ Programming – From Beginner to Beyond

Obtain Modern C++ Object-Oriented Programming (OOP) and STL skills needed for game, system, and application development.



Instructor
FRANK J. MITROPOULOS

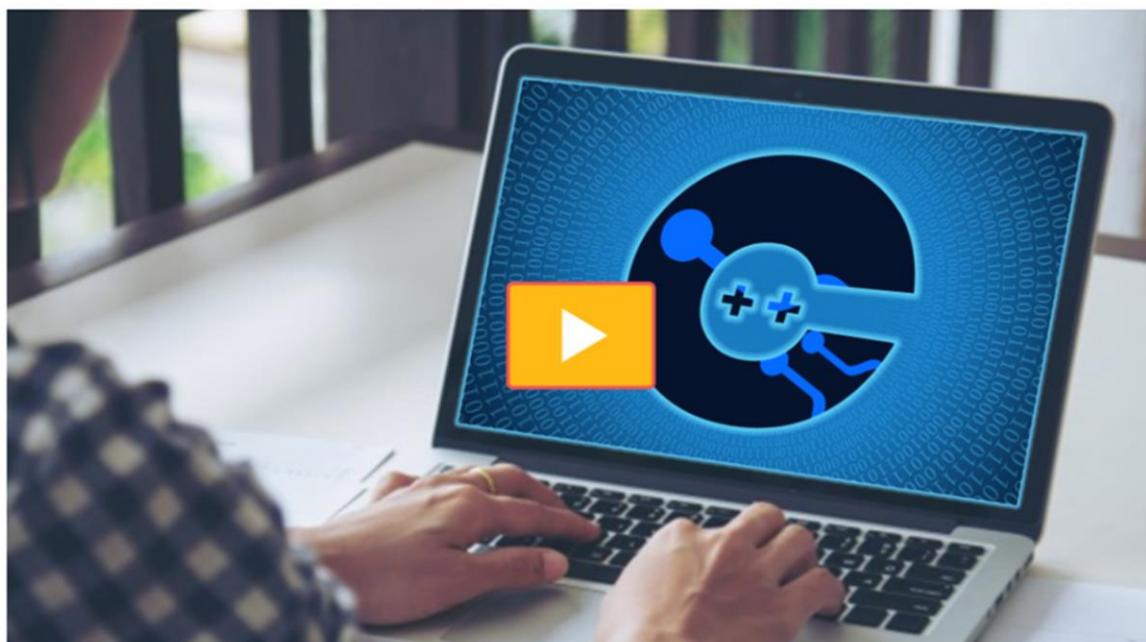
Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (13,051 REVIEWS)

TAKE THIS COURSE



Get the [C++ course here.](#)

Java SE 11 Developer 1Z0-819 OCP Course – Part 1

Getting Java Certified is great for your career. Acquire the skills to pass the Oracle Java certification exam!



Instructor
TIM BUCHALKA

Category
IT CERTIFICATION

Reviews
 (894 REVIEWS)

[TAKE THIS COURSE](#)



Get the [1Z0-819 OCP Course – Part 1 here.](#)

Android Java Masterclass – Become an App Developer

Improve your career options by learning Android app Development. Master Android Studio and build your first app today



Instructor
TIM BUCHALKA

Category
MOBILE APPS

Reviews
 (6,586 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Android Java Masterclass here.](#)

C Programming For Beginners – Master the C Language

C Programming will increase career options. Become a better dev in other languages by learning C. Pointers explained



Instructor
JASON FEDIN

Category
PROGRAMMING LANGUAGES

Reviews
★★★★★ (6,614 REVIEWS)

[TAKE THIS COURSE](#)



Get the [C Programming course here.](#)

Advanced C Programming Course

Become a True Master of the C Programming Language - Confidently Apply for Real Time or Embedded C Jobs or contracts!

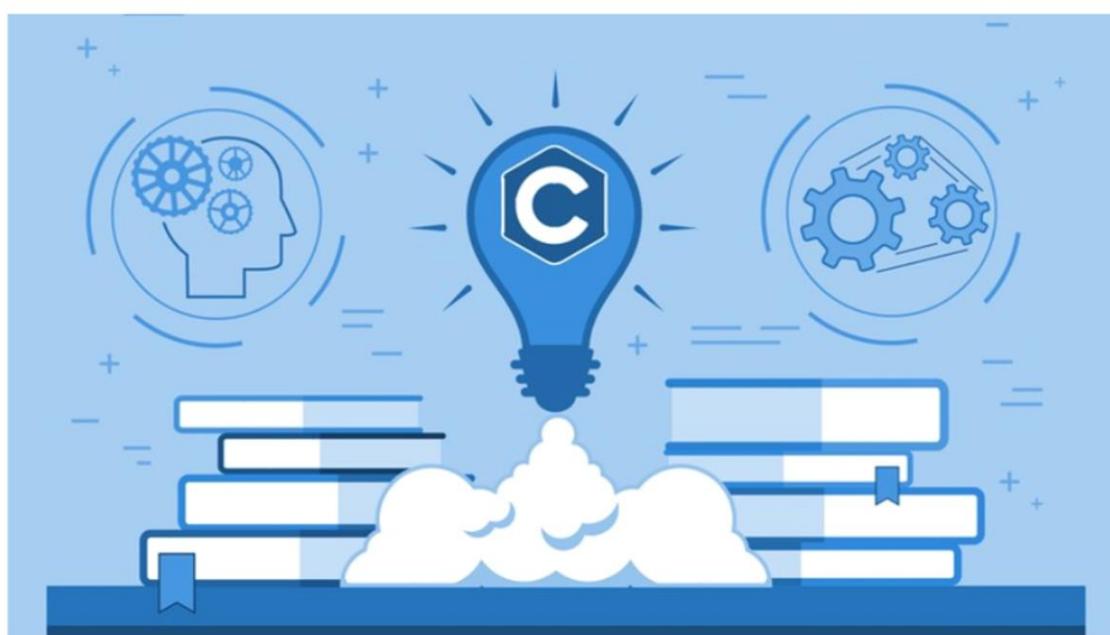


Instructor
JASON FEDIN

Category
PROGRAMMING LANGUAGES

Reviews
★★★★★ (7 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Advanced C Programming Course here](#)

Data Structures and Algorithms: Deep Dive Using Java

Learn about Arrays, Linked Lists, Trees, Hashtables, Stacks, Queues, Heaps, Sort algorithms and Search algorithms



Instructor
TIM BUCHALKA

Category
PROGRAMMING LANGUAGES

Reviews
 (3,855 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Data Structures and Algorithms course here](#)

Java Spring Tutorial Masterclass – Learn Spring Framework 5

Can't Find a good Spring Tutorial? Finally Understand Spring 5 With Spring Core, Spring MVC, Spring Boot 2 and more



Instructor
TIM BUCHALKA

Category
PROGRAMMING LANGUAGES

Reviews
 (3,707 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Java Spring Masterclass here](#)

Android App Development Masterclass using Kotlin

Learn Kotlin Android App Development And Become an Android Developer.
Incl. Kotlin Tutorial and Android Tutorial Videos

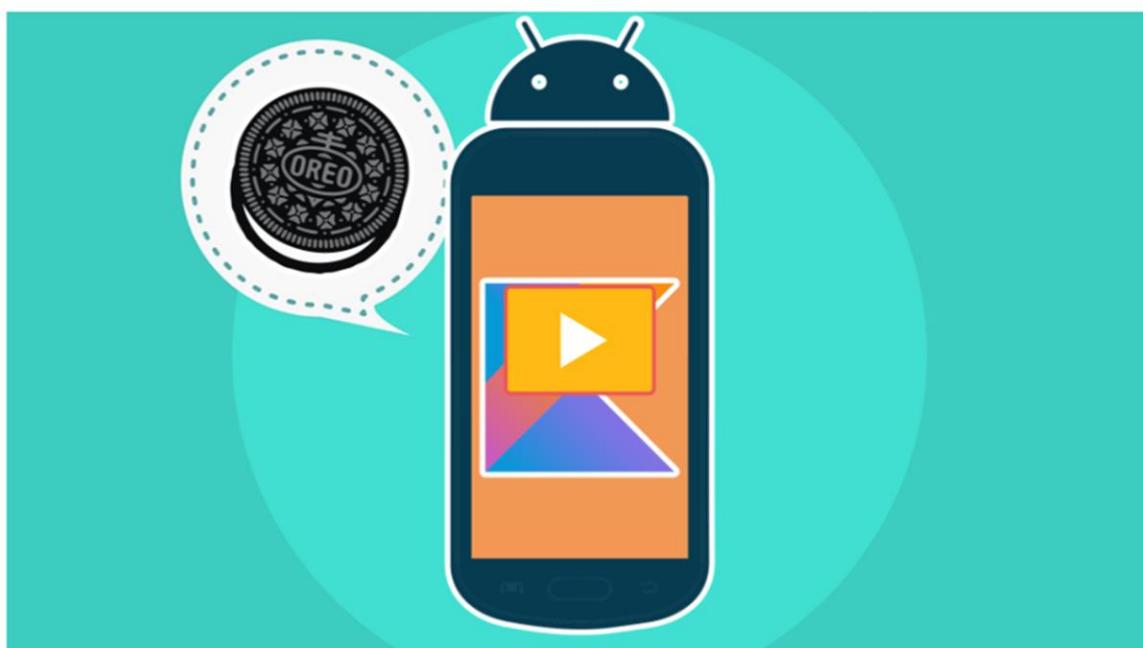


Instructor
TIM BUCHALKA

Category
MOBILE APPS

Reviews
 (1,748 REVIEWS)

[TAKE THIS COURSE](#)



[Get the Android Kotlin Masterclass here](#)

The Complete Xamarin Developer Course: iOS And Android!

Build Cross Platform Android and iOS apps with Xamarin Forms, Xamarin Classic, Azure Mobile App Services, Rest and more



Instructor
EDUARDO ROSAS

Category
MOBILE APPS

Reviews
 (2,008 REVIEWS)

[TAKE THIS COURSE](#)



[Get the Complete Xamarin course here.](#)

Oracle Java Certification – Pass the Associate 1Z0-808 Exam.

This course will help you learn the steps to becoming an Oracle Certified Associate (OCA) and get a higher paying job!



Instructor
GORAN LOCHERT

Category
IT CERTIFICATION

Reviews
 (2,046 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Oracle Java Certification course here.](#)

Master MATLAB through Guided Problem Solving

Become an expert in MATLAB Programming and Scientific Computing.
Advance your career in Engineering Physics Biology etc

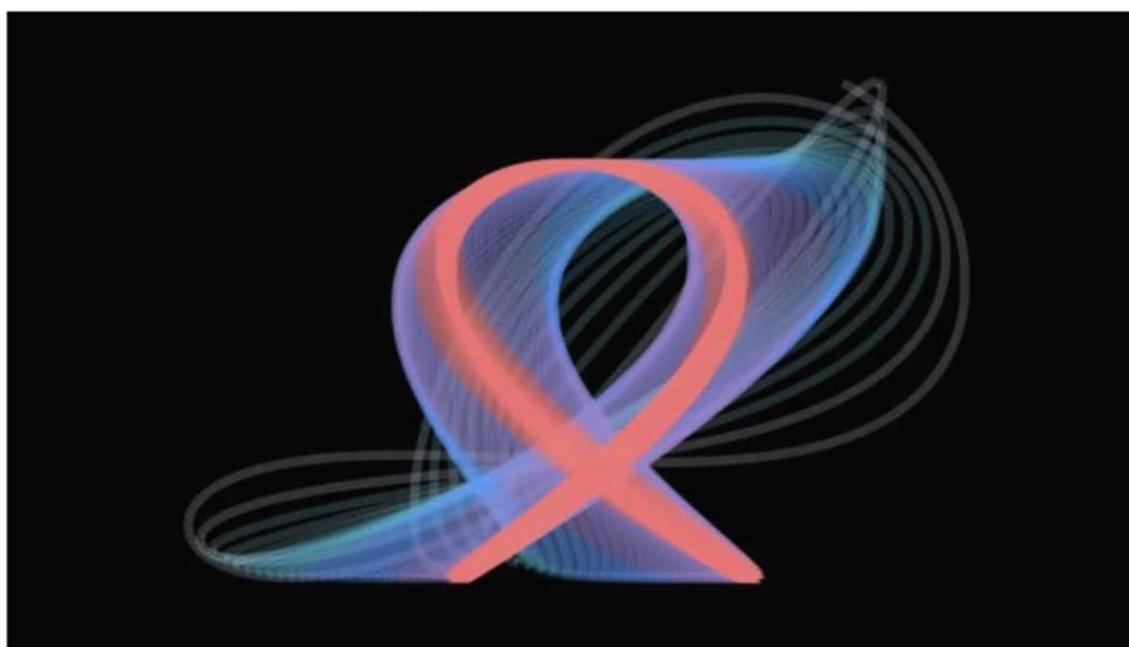


Instructor
MICHAEL COHEN

Category
PROGRAMMING LANGUAGES

Reviews
 (1,453 REVIEWS)

[TAKE THIS COURSE](#)



Get the [MATLAB course here](#)

Kotlin for Java Developers

Use your Java skills to learn Kotlin fast. Enhance career prospects and master Kotlin, including Java interoperability



Instructor
TIM BUCHALKA

Category
PROGRAMMING LANGUAGES

Reviews
 (1,149 REVIEWS)

TAKE THIS COURSE



Get the [Kotlin for Java Developers course here.](#)

SQL for Beginners: Learn SQL using MySQL and Database Design

Understand SQL using the MySQL database. Learn Database Design and Data Analysis with Normalization and Relationships



Instructor
JON AVIS

Category
DATABASES

Reviews
 (1,590 REVIEWS)

TAKE THIS COURSE



Get the [SQL course here.](#)

Windows Presentation Foundation Masterclass

Leverage WPF with C# and XAML to build real world skills with Azure, REST, MVVM and Machine Learning



Instructor
EDUARDO ROSAS

Category
PROGRAMMING LANGUAGES

Reviews
 (1,017 REVIEWS)

[TAKE THIS COURSE](#)



Get the [WPF Masterclass here.](#)

Python REST APIs with Flask, Docker, MongoDB, and AWS DevOps

Learn Python coding with RESTful API's using the Flask framework. Understand how to use MongoDB, Docker and Tensor flow.



Instructor
EL FAROUK YASSER

Category
WEB DEVELOPMENT

Reviews
 (602 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Python APIs course here.](#)

Learn Java Programming Crash Course

This Java For Beginners Course Includes Java Basics And Core Java Skills Training To Make You A Software Developer Fast



Instructor
TIM BUCHALKA

Category
PROGRAMMING LANGUAGES

Reviews
 (721 REVIEWS)

[TAKE THIS COURSE](#)



[Get the Java crash course here.](#)

PHP for Beginners

Build a Content Management System from Scratch with PHP and MySQL



Instructor
DAVE HOLLINGSWORTH

Category
PROGRAMMING LANGUAGES

Reviews
 (608 REVIEWS)

[TAKE THIS COURSE](#)



[Get the PHP course here.](#)

Machine Learning with Python from Scratch

Mastering Machine Learning Algorithms including Neural Networks with Numpy, Pandas, Matplotlib, Seaborn and Scikit-Learn



Instructor
CARLOS QUIROS

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (198 REVIEWS)

TAKE THIS COURSE



Get the [Machine Learning - Python course here.](#)

What's New in Java 9 – Modules and More!

Improve Your Career Prospects by Learning About New Java 9 Features Like Modules, JShell, Processes and More.



Instructor
FRANK J. MITROPOULOS

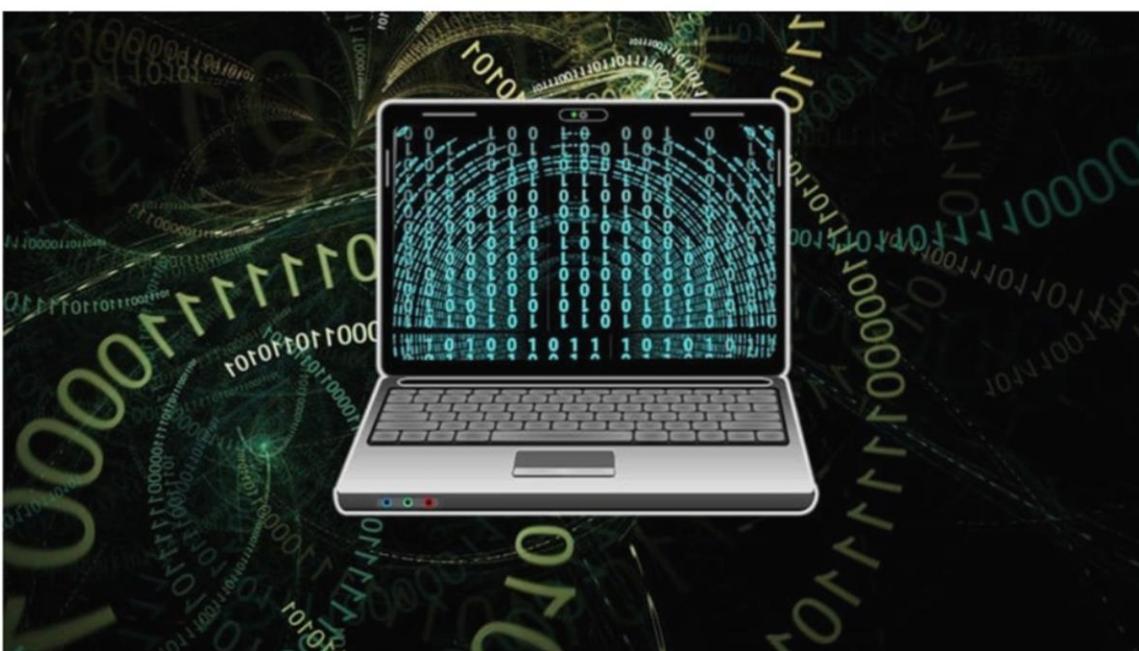
Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (385 REVIEWS)

TAKE THIS COURSE



Get the [Java 9 course here.](#)

Java Enterprise Edition 8 for Beginners course

Understand Jakarta EE, JPA, CDI, JAX-RS, REST, JWT, JSON-P and JSON-B and more. Add "JEE Developer" to your résumé!



Instructor
LUQMAN SAEED

Category
WEB DEVELOPMENT

Reviews
 (376 REVIEWS)

[TAKE THIS COURSE](#)



Get the [JEE8 course here.](#)

Ethical Hacking Course: Protect Yourself From Being Hacked

Learn about the Dark Web, Social Engineering, Backdoors, Website Hacking, SQL Injection, Wireless attacks and more!



Instructor
ATİL SAMANCIÖĞLU

Category
NETWORK & SECURITY

Reviews
 (215 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Ethical Hacking course here.](#)

The Java Design Patterns Course

Understand the how and the why of the gang of four design patterns using Java.



Instructor
JASON FEDIN

Category
PROGRAMMING LANGUAGES

Reviews
 (251 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Java Design Patterns course here.](#)

Android LibGDX Game Development Masterclass

Become a real games programmer. Create Games Using Java with the LibGDX Game Development Framework.



Instructor
GORAN LOCHERT

Category
GAME DEVELOPMENT

Reviews
 (335 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Android LibGDX course here.](#)

Advanced Algorithms in Java

Understand Algorithms and Data structure at a deep level. Grow your career and be ready to answer interview questions!



Instructor
MARCOS COSTA

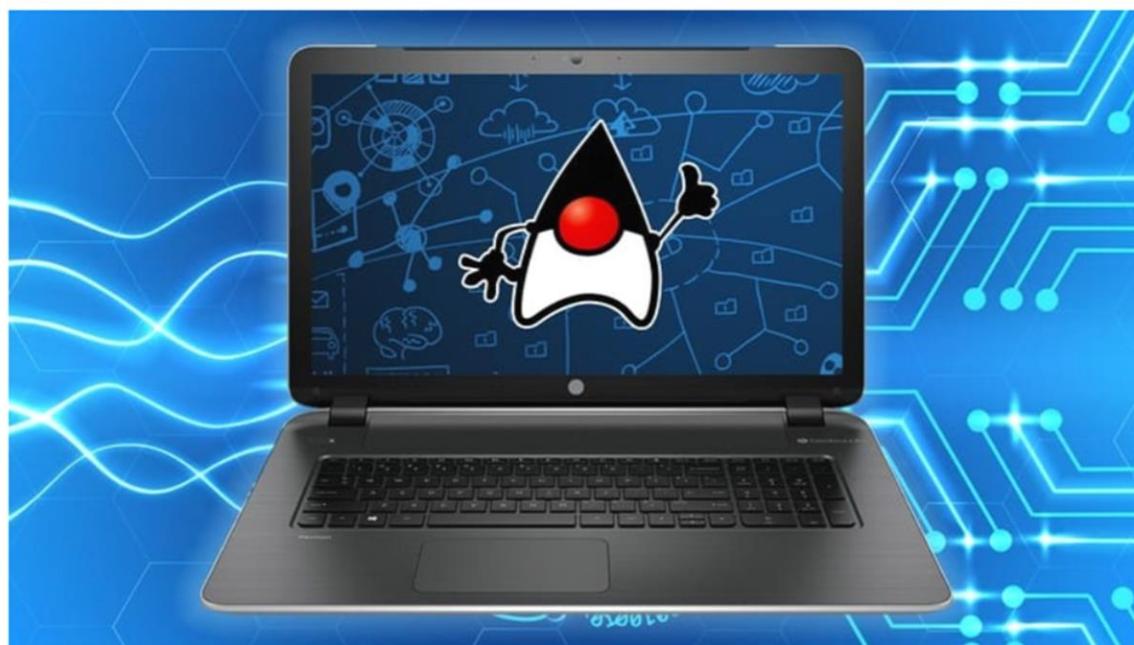
Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (65 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Advanced Algorithms - Java course here.](#)

Git & GitHub Masterclass

Add real world development team skills for version control and source control to your resume & programming arsenal!



Instructor
EDUARDO ROSAS

Category

DEVELOPMENT TOOLS

Reviews

★★★★★ (169 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Git & GitHub Masterclass here.](#)

The Advanced Xamarin Developer Masterclass

Includes Xamarin University Quality content - This Advanced Xamarin Tutorial Course Focuses on Cross Platform Concepts



Instructor
EDUARDO ROSAS

Category
WEB DEVELOPMENT

Reviews
 (135 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Advanced Xamarin Masterclass here.](#)

3D Programming with JavaScript and the Three.js 3D Library

Create 3D computer graphics, using webgl in a cross-browser environment. Learn about 3D Graphical space, and 3D Depth.

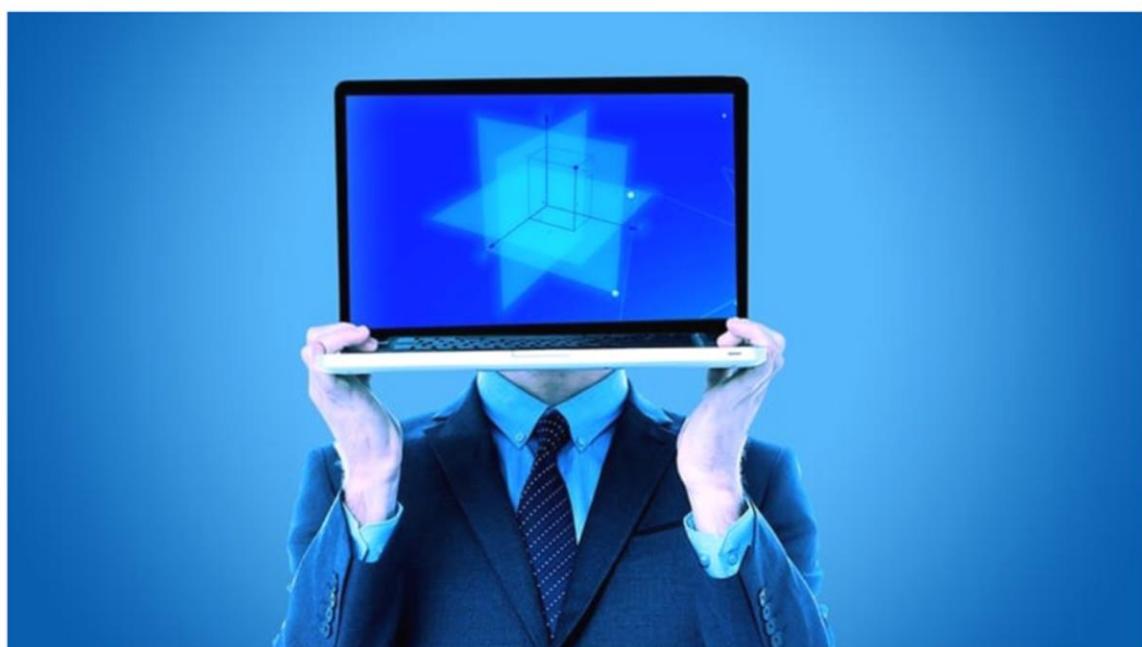


Instructor
SHAY TAVOR

Category
WEB DEVELOPMENT

Reviews
 (262 REVIEWS)

[TAKE THIS COURSE](#)



Get the [3D Programming course here.](#)

Learn C# for Beginners Crash Course

Obtain C# Programming Language Skills With This C# Tutorial. Acquire Essentials Skills To Get a C# Developer Job Today.



Instructor
TIM BUCHALKA

Category
PROGRAMMING LANGUAGES

Reviews
 (266 REVIEWS)

TAKE THIS COURSE



Get the [C# crash course here.](#)

Python Tkinter Masterclass – Learn Python GUI Programming

Build Python Tkinter Desktop Applications



Instructor
VOLKAN ATIŞ

Category
PROGRAMMING LANGUAGES

Reviews
 (188 REVIEWS)

TAKE THIS COURSE



Get the [Python Tkinter Masterclass here.](#)

Rust Programming Language for Beginners

More effective than C++. Develop your own Rust Programming library and increase your career options.



Instructor
DIWAKAR SINGH

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (108 REVIEWS)

TAKE THIS COURSE



Get the [Rust course here.](#)

ArcPy for Python Developers using ArcGIS Pro

GIS is hot - take your python skills to new levels and greatly increase your career options.



Instructor
GRAEME BROWNING

Category

WEB DEVELOPMENT

Reviews

★★★★★ (184 REVIEWS)

TAKE THIS COURSE



Get the [ArcPy for Python course here.](#)

Kotlin LibGDX Game Developers Masterclass

Become a real games programmer. Create Games Using Kotlin with the LibGDX Game Development Framework.



Instructor
GORAN LOCHERT

Category
GAME DEVELOPMENT

Reviews
 (101 REVIEWS)

TAKE THIS COURSE



[Get the Kotlin LibGDX Masterclass here.](#)

Unity Game Developers Masterclass: Write Games using C#

Create and Extend your own games. Learn scenes, user interfaces, physics, game logic, scripting, and custom tools.

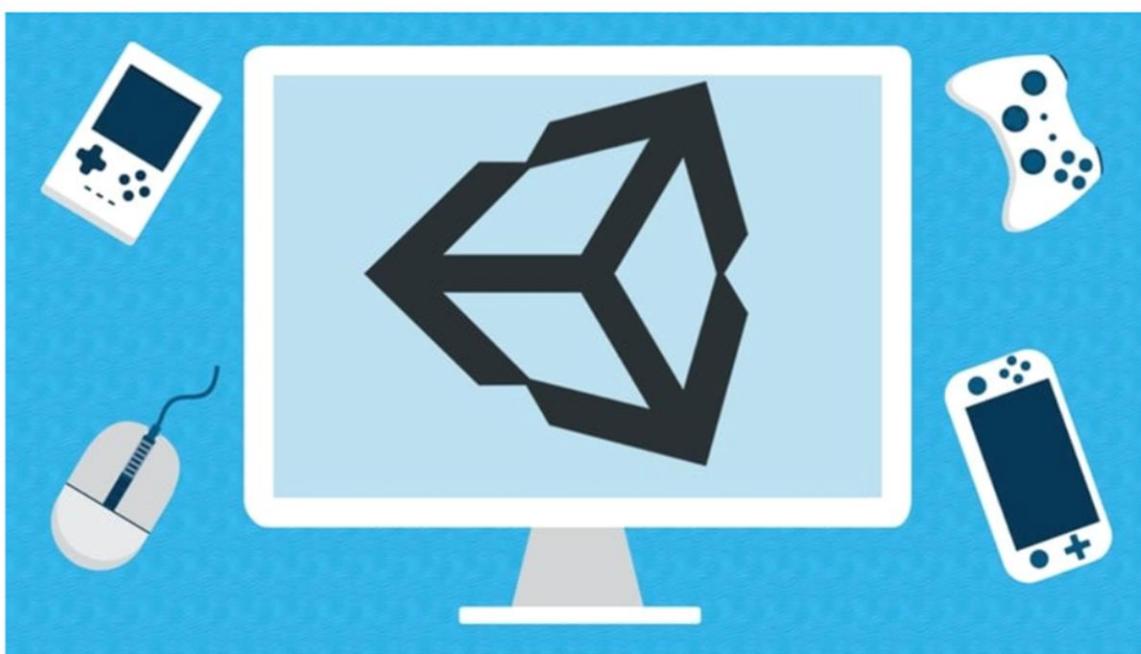


Instructor
GORAN LOCHERT

Category
GAME DEVELOPMENT

Reviews
 (57 REVIEWS)

TAKE THIS COURSE



[Get the Unity Masterclass here.](#)

Python GUI Programming Using PyQt5

Build Python GUI Desktop Applications With PYQT and Master Sqlite



Instructor
VOLKAN ATIS

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (41 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Python GUI - PyQt5 course here.](#)

Ruby on Rails for Beginners

Create real world applications using this in demand Web Development framework.



Instructor

STEPHEN CHESNOWITZ

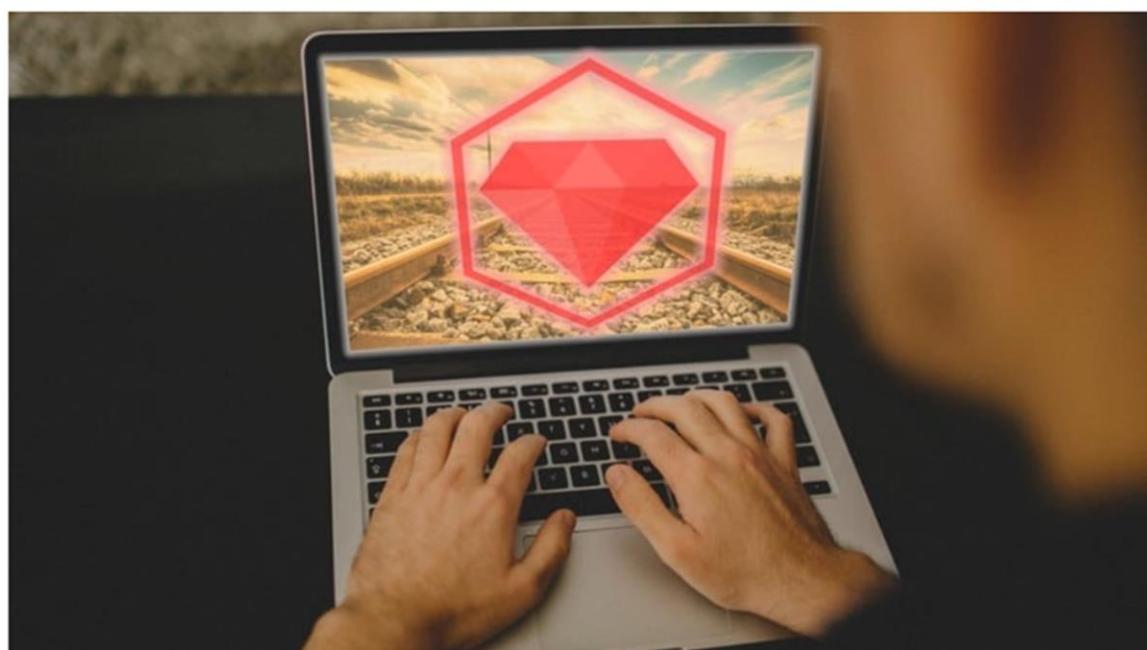
Category

WEB DEVELOPMENT

Reviews

★★★★★ (48 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Ruby on Rails course here.](#)

Search Algorithms in Artificial Intelligence with Java

This Artificial Intelligence Course Teaches Theory, Implementation, and Applications With Robot Path Planning



Instructor

DR. SEYEDALI MIRJALILI

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (25 REVIEWS)

[TAKE THIS COURSE](#)

Get the [Search Algorithms in AI course here.](#)

The Complete Javascript Course for Developers

Learn JavaScript Quickly. This JavaScript Class Will Teach You JavaScript Fundamentals And Is Beginner Friendly



Instructor

CHARLES E. BROWN

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (82 REVIEWS)

[TAKE THIS COURSE](#)

Get the [Javascript course here.](#)

Azure Machine Learning using Cognitive Services

Learn the Azure Machine Learning Studio, Azure Bot Service, Video Indexing service, Computer Vision for OCR and more!



Instructor
EDUARDO ROSAS

Category
WEB DEVELOPMENT

Reviews
 (47 REVIEWS)

TAKE THIS COURSE



Get the [Azure Machine Learning course here.](#)

Mastering IntelliJ IDEA and Android Studio

Instantly Become More Productive, Learn To Get The Most Out Of IntelliJ IDEA and Android Studio



Instructor
ROBERT GIOIA

Category
DEVELOPMENT TOOLS

Reviews
 (40 REVIEWS)

TAKE THIS COURSE



Get the [IntelliJ and Android Studio course here.](#)

Ruby for Beginners

Ruby Programming Skills are vital for Ruby on Rails development. These tutorials will teach you Ruby fast!



Instructor

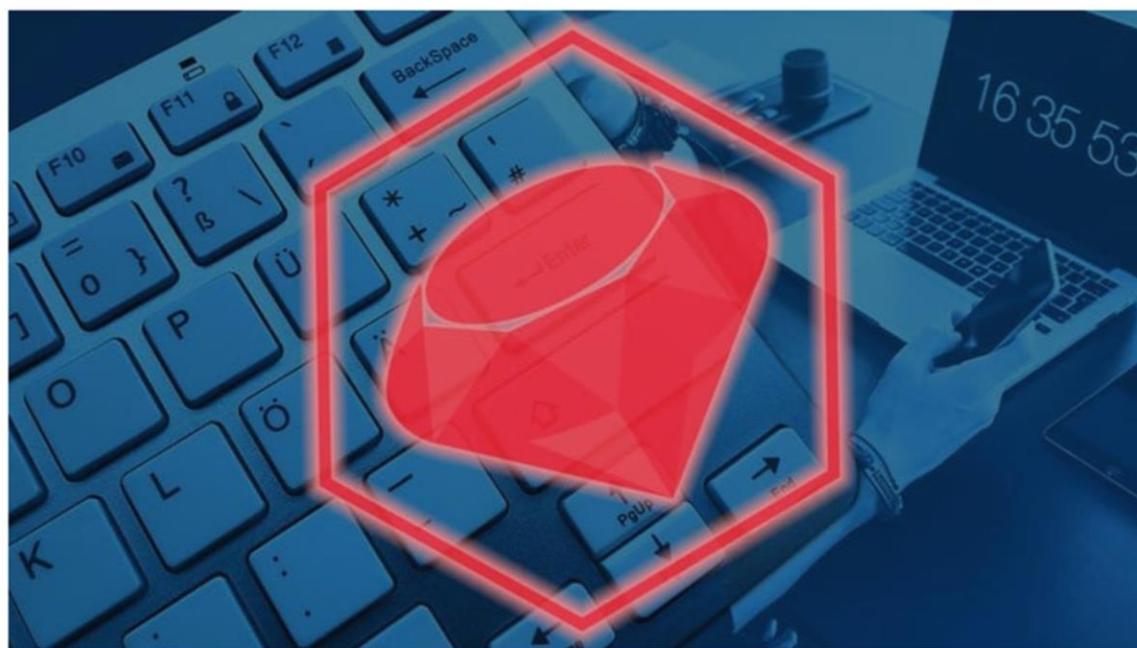
STEPHEN CHESNOWITZ

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (58 REVIEWS)

[TAKE THIS COURSE](#)

[Get the Ruby for Beginners course here.](#)

Learning Bootstrap – From HTML to WordPress Theme

Design and develop static websites, and convert them to dynamic Websites using HTML, CSS, Bootstrap and jQuery!



Instructor

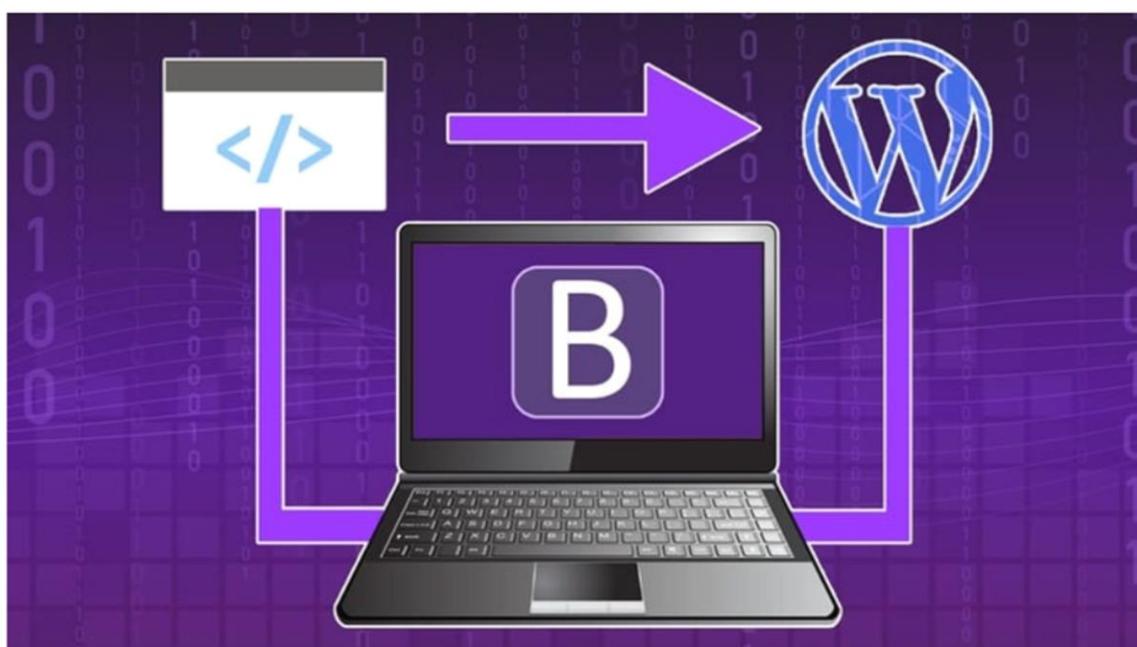
ABUL HOSSAIN

Category

WEB DEVELOPMENT

Reviews

★★★★★ (34 REVIEWS)

[TAKE THIS COURSE](#)

[Get the Learning Bootstrap course here.](#)

HTML and CSS Masterclass

Web development require these skills. Learn HTML5 and CSS 3 as well a Bootstrap and XHTML!

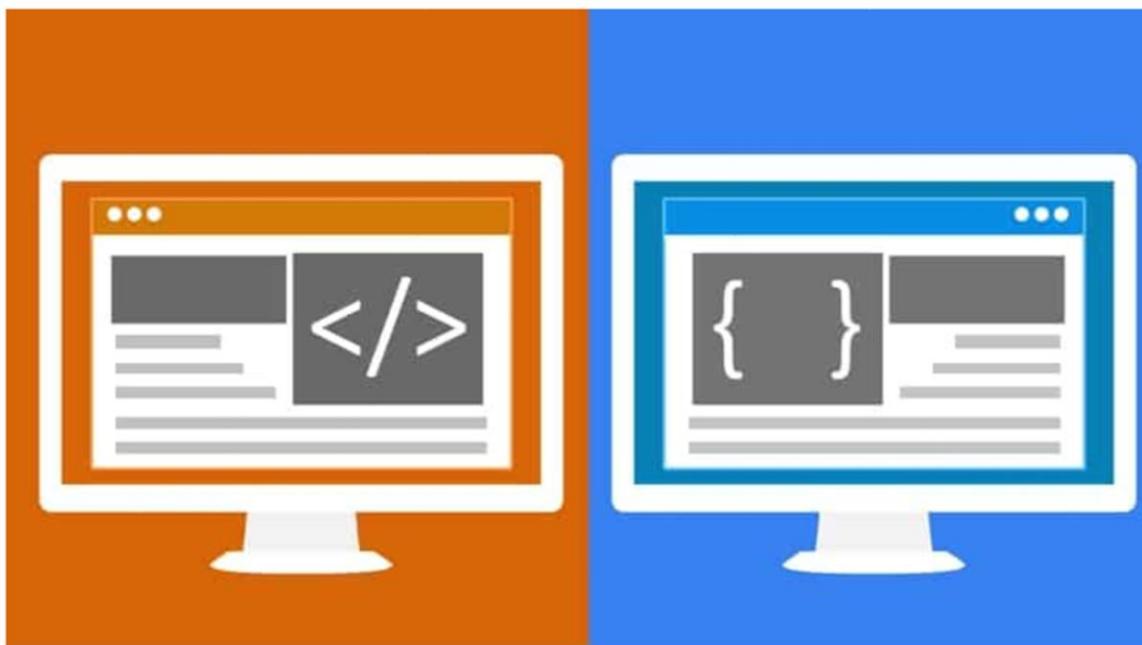


Instructor
ROBERT GIOIA

Category
WEB DEVELOPMENT

Reviews
★★★★★ (40 REVIEWS)

TAKE THIS COURSE



Get the [HTML and CSS Masterclass here.](#)

Android Firebase Masterclass – Master Google Firebase

Create Cloud based Android applications using Google Firebase and expand your career options

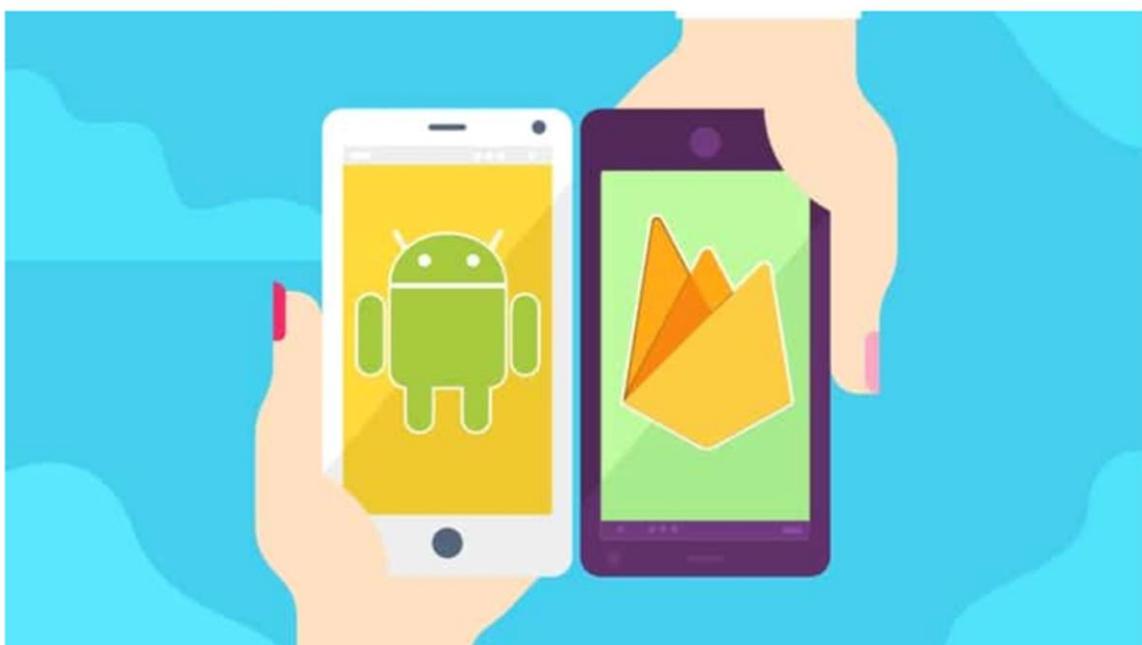


Instructor
JASON FEDIN

Category
MOBILE APPS

Reviews
★★★★★ (8 REVIEWS)

TAKE THIS COURSE



Get the [Android Firebase Masterclass here.](#)

iOS Firebase Masterclass – Real time Database and Firestore

Learn Authentication, Cloud Functions, Crashytics, Ads, a Multiplayer game, Push notifications and more.



Instructor
KEVIN MURPHY

Category
MOBILE APPS

Reviews
 (0 REVIEW)

TAKE THIS COURSE



Get the [iOS Firebase Masterclass here.](#)

Master CI/CD for Xamarin

Learn Continuous Integration and Continuous Deployment / Delivery to increase your employability in 2019 and beyond!



Instructor
EDUARDO ROSAS

Category
SOFTWARE ENGINEERING

Reviews
 (3 REVIEWS)

TAKE THIS COURSE



Get the [CI/CD for Xamarin here](#)

Learn Go for Beginners Crash Course (Golang)

Master the Go Programming Language Step by Step - No previous programming experience required.



Instructor

TREVOR SAWLER

Category

PROGRAMMING LANGUAGES

Reviews

★★★★★ (42 REVIEWS)

[TAKE THIS COURSE](#)

Get the [Go Crash Course here](#)

Master CI/CD for Android Developers

Learn Continuous Integration and Continuous Deployment / Delivery to increase your employability in 2019 and beyond!



Instructor

EDUARDO ROSAS

Category

SOFTWARE ENGINEERING

Reviews

★★★★★ (0 REVIEW)

[TAKE THIS COURSE](#)

Get the [CI/CD for Android here](#)

Master CI/CD for iOS Developers

Learn Continuous Integration and Continuous Deployment / Delivery to increase your employability in 2019 and beyond!



Instructor
EDUARDO ROSAS

Category
SOFTWARE ENGINEERING

Reviews
 (0 REVIEW)

[TAKE THIS COURSE](#)



[Get the CI/CD for iOS Developers here](#)

Master CI/CD for React Native

Learn Continuous Integration and Continuous Deployment / Delivery to increase your employability in 2019 and beyond!



Instructor
EDUARDO ROSAS

Category
SOFTWARE ENGINEERING

Reviews
 (0 REVIEW)

[TAKE THIS COURSE](#)



[Get the CI/CD for React Native here](#)

Java SE 11 Developer 1Z0-819 OCP Course – Part 2

Pass Oracle's 1Z0-819 exam and become Java certified. Get skilled in everything you need including the tricky stuff.



Instructor
TIM BUCHALKA

Category
IT CERTIFICATION

Reviews
 (5 REVIEWS)

[TAKE THIS COURSE](#)



Get the [1Z0-819 OCP Course – Part 2 here.](#)

Introduction to Continuous Integration & Continuous Delivery

Find out why CI and CD, coupled with DevOps will give you a competitive advantage over developers without this knowledge

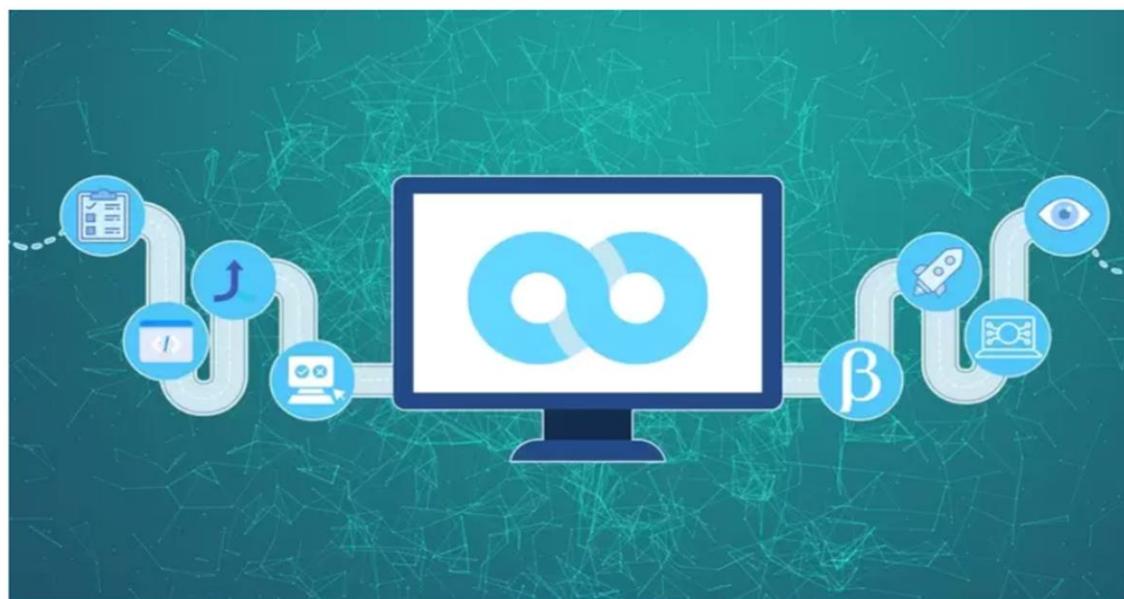


Instructor
TIM BUCHALKA

Category
DEVELOPMENT TOOLS

Reviews
 (3 REVIEWS)

[TAKE THIS COURSE](#)



Get the [Introduction to CI/CD here.](#)

CHAPTER 5 – LEARN TO CODE COURSE



I've been busy working on a Learn to Code course. The course is the only course of mine that is not available on Udemy but is available to **you for free**.

You might be wondering why am I releasing another Learn to Code Course, why is it not on Udemy, and what's different about the course compared to a course like my Java Masterclass?

Good questions, all of them.

Firstly, I am a believer in students learn to program having an understanding of how a computer works under the hood.

For example, did you know that a programming language you use like Java to create programs makes no sense to a computer?

The computer has no understanding of Java as such. It can only understand a single language. That language is called machine code. Machine code uses binary – 0's and 1's and makes sense to computers.

Any modern programming language you use has to be “converted” into machine code before a computer can run it.

I think that you learning how a computer runs programs you write, and what binary is, how to use it, and so on really will give you an edge in your programming career.

That's why I created this Learn to Code course - it's designed to teach you things like this that very few other courses go into, but that I feel will help you.

That's not all I cover, and by the end of the course, you will have worked with C++, Java, Kotlin, and Python and will truly understand how this stuff works, but also other programming concepts.

Think of this as a supplement course to any of your other training courses, including all of the Learn Programming Academy courses.

A publisher contacted me about the course wanting to publish it as a paid course.

But I refused. Instead, I'm making it 100% free for you to access right now.

It's on YouTube, click on the link below to watch the first video. Be sure to subscribe to be notified each time I release the next video in the series.

You will find this course to be fun and informative, so be sure to check it out.

I'm releasing a video each week for this course.

Click [here](#) to watch the first Learn to Code video on YouTube now (there are already several videos available for immediate viewing).



CHAPTER 6 – PROGRAMMING TIP OF THE DAY VIDEOS

At last count, there are 106 videos available for you to freely access. They cover a range of important topics such as

- How long does it take to become a programmer?
- Do you need a degree to get a programming job?
- What is the number one skill you need to possess to be an effective programmer?
- Are you too old to become a programmer?
- Do you need to be good at math to become a programmer?
- But other questions like which is better to learn Kotlin or Java, What is JEE, and other related videos.

I am sure these videos will be very useful for you.

Each video is available for you to watch - the video image in the PDF contains the question that I answer in each video - click the image to play each video or the link below each video.

Click an image to watch that video right now:



[Watch Programming Tips video here](#)

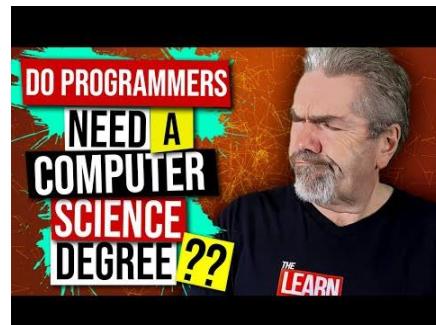


[Watch Timeframes video here](#)

Click an image to watch that video right now:



[Watch Careers video here](#)



[Watch Degree video here](#)



[Watch Skills video here](#)



[Watch Am I Too Old video here](#)



[Watch Math video here](#)



[Watch Interview Tips video here](#)



[Watch Languages video here](#)



[Watch Transitioning video here](#)



[Watch Search Engines video here](#)



[Watch First Job video here](#)

Click an image or the link below it to watch that video right now:



[Watch Technology video here](#)



[Watch Career video here](#)



[Watch Overwhelmed video here](#)



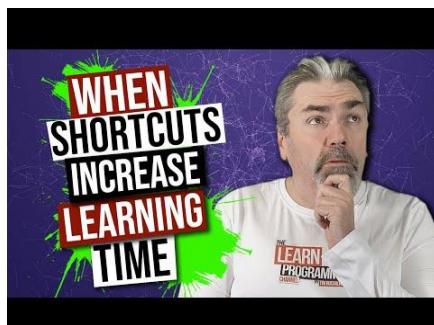
[Watch Practice Skills video here](#)



[Watch Java or Kotlin video here](#)



[Watch Training video here](#)



[Watch Shortcuts video here](#)



[Watch Courses video here](#)



[Watch AI video here](#)



[Watch Code Fear video here](#)

Click an image to watch that video right now:



[Watch Better Programmer video here](#)



[Watch Course Completed video here](#)



[Watch Job Ready video here](#)



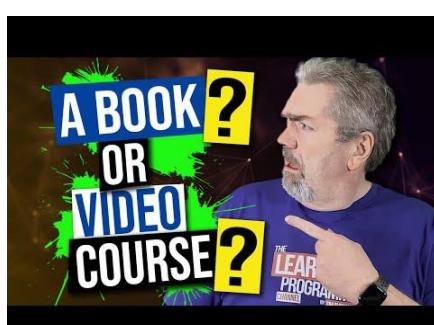
[Watch Move video here](#)



[Watch Engineer vs Dev video here](#)



[Watch Book Supplement video here](#)



[Watch Book or Video video here](#)



[Watch CSharp Vs. Java video here](#)



[Watch Freelance video here](#)



[Watch Apply for Job video here](#)

Click an image to watch that video right now:



[Watch Framework video here](#)



[Watch Coding Tests video here](#)



[Watch Linux Skills video here](#)



[Watch Certifications video here](#)



[Watch Design Patterns video here](#)



[Watch Solving Problems video here](#)



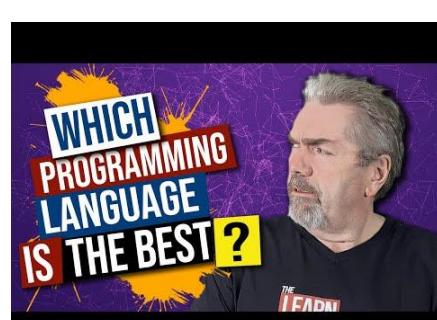
[Watch Right Career video here](#)



[Watch Online Course video here](#)



[Watch API video here](#)



[Watch Best Language video here](#)

Click an image to watch that video right now:



[Watch All Languages video here](#)



[Watch Distractions video here](#)



[Watch IDE video here](#)



[Watch Courses video here](#)



[Watch Jobs In Demand video here](#)



[Watch Requirements video here](#)



[Watch Resume video here](#)



[Watch Two Areas video here](#)



[Watch Earn Money video here](#)

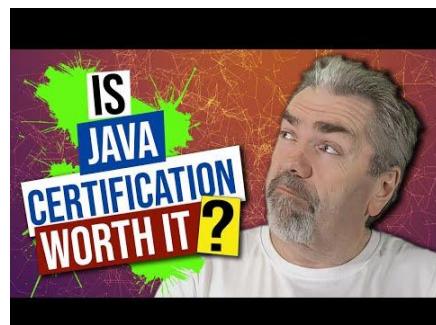


[Watch Best Way video here](#)

Click an image to watch that video right now:



[Watch Goal Setting video here](#)



[Watch Certification video here](#)



[Watch Challenges video here](#)



[Watch Smarts video here](#)



[Watch Enroll video here](#)



[Watch Responsibility video here](#)



[Watch Questions video here](#)



[Watch Hardware video here](#)



[Watch Instructors video here](#)



[Watch How Long It Takes video here](#)

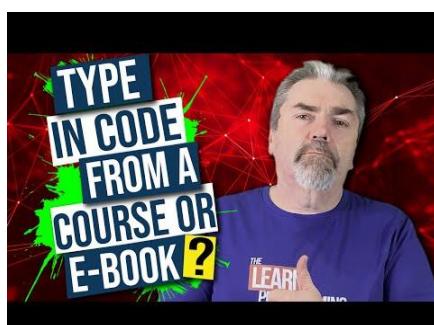
Click an image to watch that video right now:



[Watch Self-Reliance video here](#)



[Watch Software video here](#)



[Watch Type Code video here](#)



[Watch Opinions video here](#)



[Watch Python video here](#)



[Watch Kotlin First video here](#)



[Watch Java vs JEE video here](#)



[Watch Big Apps video here](#)



[Watch Languages video here](#)



[Watch Ways to Improve video here](#)

Click an image to watch that video right now:



[Watch Descriptions video here](#)



[Watch Flutter video here](#)



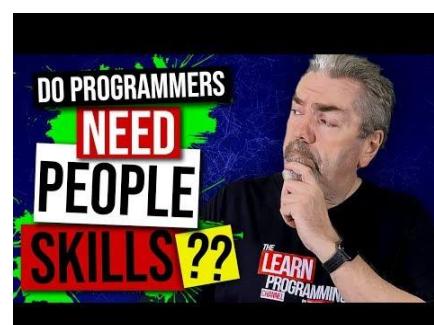
[Watch 3 Things video here](#)



[Watch JDK, JVM, JRE video here](#)



[Watch Tips video here](#)



[Watch People Skills video here](#)



[Watch Designer vs. Dev video here](#)



[Watch Learn Coding video here](#)



[Watch Stages video here](#)



[Watch Learn Programming video here](#)

Click an image to watch that video right now:



[Watch Front-End video here](#)



[Watch Back-End video here](#)



[Watch Full-Stack video here](#)



[Watch Algorithms video here](#)



[Watch Pay or Free video here](#)



[Watch IDE or CMD video here](#)



[Watch Hardware video here](#)



[Watch Impostor Syndrome video here](#)



[Watch Job Specs video here](#)



[Watch Focus and Plan video here](#)

Click an image to watch that video right now:



[Watch Problems video here](#)



[Watch 2 Languages video here](#)



[Watch Interview Tips video here](#)



[Watch Junior Dev video here](#)



[Watch Best Practices video here](#)



[Watch Comm Skills video here](#)



[Watch Roles video here](#)



[Watch Competitive video here](#)



[Watch Persistence video here](#)



[Watch Key to Success video here](#)

Click an image to watch that video right now:



[Watch JDK Version video here](#)



[Watch JVM-JDK video here](#)



[Watch Why Fail video here](#)



[Watch Questions video here](#)