1 implement\_backend\_of\_compiler

Source code :

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#include <string.h>

void main()

{

    char icode[10][30], str[20], opr[10];

    int i = 0;

    printf("Enter the set of intermediate code (terminate by 'exit'): ");

    do

    {

        scanf("%s", icode[i]);

    } while (strcmp(icode[i++], "exit") != 0);

    printf("\nTarget code generation");

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

    i = 0;

    do

    {

        strcpy(str, icode[i]);

        switch (str[3])

        {

        case '+':

            strcpy(opr, "ADD");

            break;

        case '-':

            strcpy(opr, "SUB");

            break;

        case '\*':

            strcpy(opr, "MUL");

            break;

        case '/':

            strcpy(opr, "DIV");

            break;

        default:

            break;

        }

        printf("\n\tMOV %c,R%d", str[2],i);

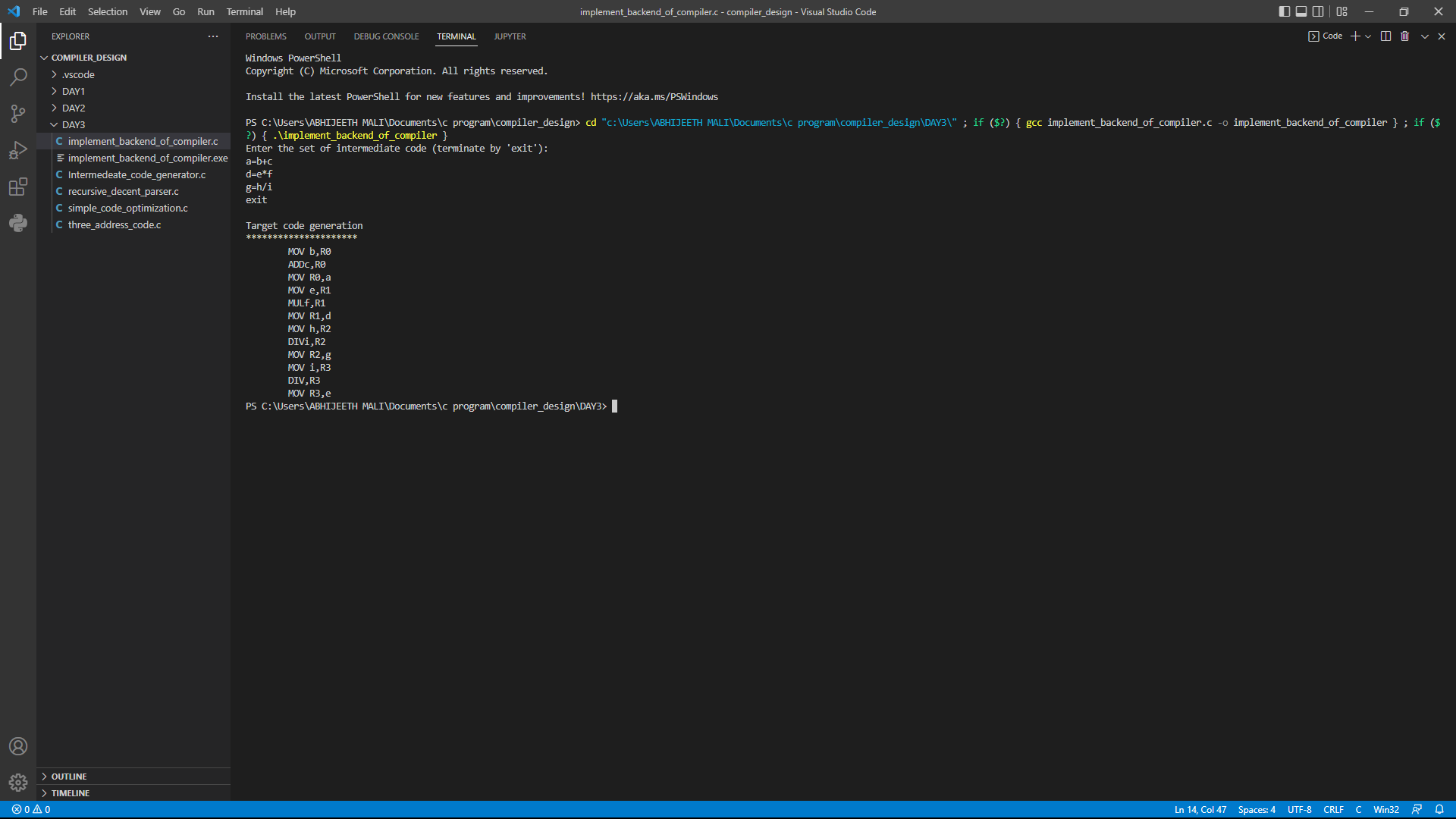
        printf("\n\t%s%c,R%d", opr, str[4], i);

        printf("\n\tMOV R%d,%c", i, str[0]);

    } while (strcmp(icode[i++], "exit") != 0);

}

Output :



2. Intermedeate\_code\_generator

Source code :

#include <stdio.h>

#include <string.h>

#include <conio.h>

struct op

{

    char l;

    char r[20];

} op[10], pr[10];

int main()

{

    int n, i, j, z = 0;

    char temp;

    char \*p;

    printf("Enter the no of values : ");

    scanf("%d", &n);

    for (i = 0; i < n; i++)

    {

        printf("\tLeft : \t");

        op[i].l = getche();

        printf("\tRight : \t");

        scanf("%s", op[i].r);

    }

    printf("Intermediate Code\n\n");

    for (i = 0; i < n; i++)

    {

        printf("%c=", op[i].l);

        printf("%ss\n", op[i].r);

    }

    for (i = 0; i < n; i++)

    {

        temp = op[i].l;

        for (j = 0; j < n; j++)

        {

            p = strchr(op[i].r, temp);

            if (p)

            {

                pr[z].l = op[i].l;

                strcpy(pr[z].r, op[i].r);

                z++;

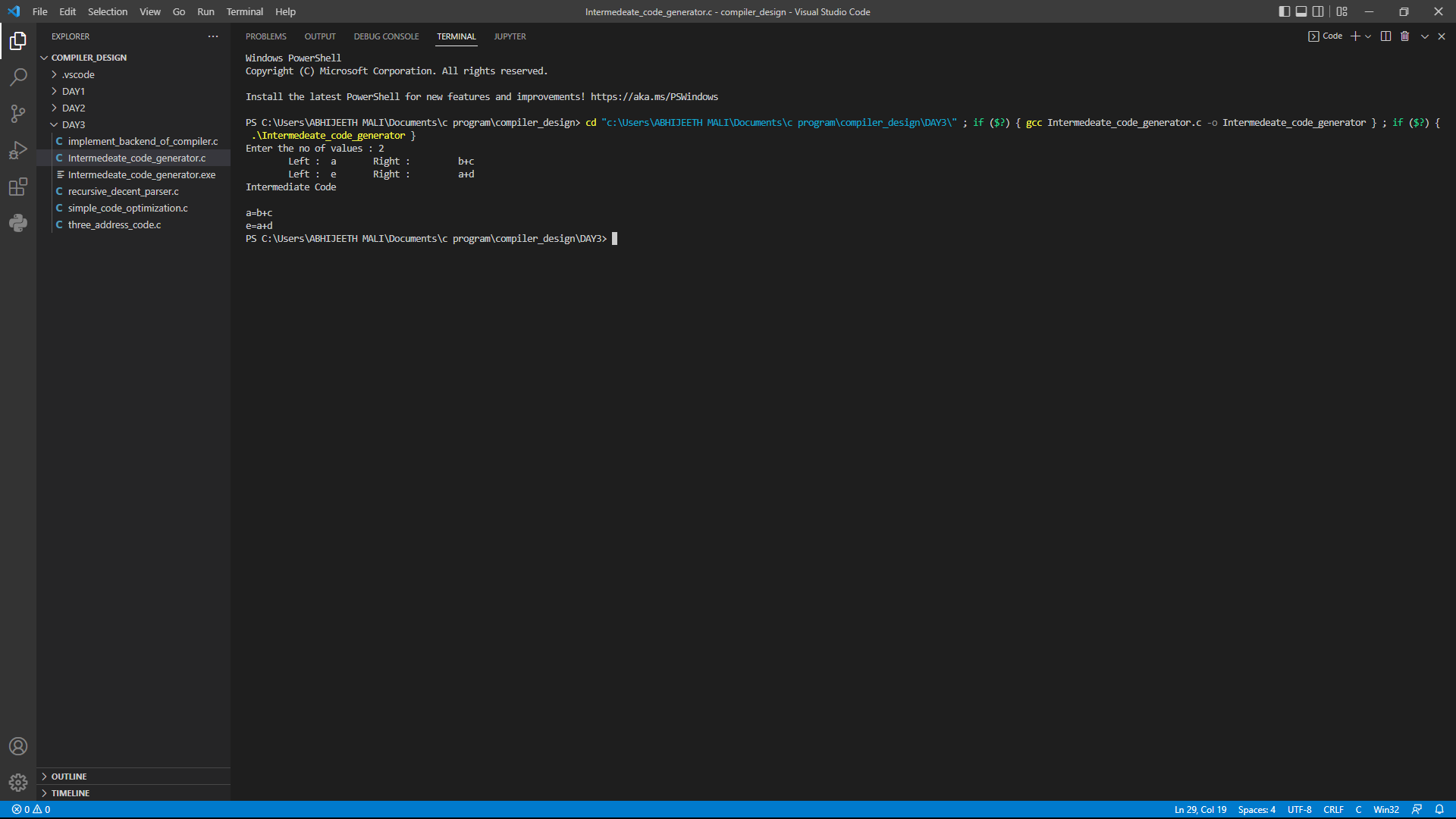
            }

        }

    }

}

Output :



3. recursive\_decent\_parser

Source code :

// S --> (L)/a

// L --> SL`

// L'--> ,SL'/e

#include <stdio.h>

#include <string.h>

int S();

int Ldash();

int L();

char \*ip;

char string[50];

int main()

{

    printf("Enter the string : ");

    scanf("%s",&string);

    ip = string;

    printf("\n-------------------------------------------------------------");

    printf("\n\nInput\t\tAction\n\n");

    if (S())

    {

        printf("\n-------------------------------------------------------------");

        printf("\nString is Successfully Parsed\n\n\n");

    }

    else

    {

        printf("\n-------------------------------------------------------------");

        printf("\nError in parsing string\n");

    }

}

int L()

{

    printf("%s\t\tL-->SL' \n",ip);

    if (S())

    {

        if(Ldash())

        {

            return 1;

        }

        else

        {

            return 0;

        }

    }

    else

    {

        return 0;

    }

}

int Ldash()

{

    if(\*ip ==',')

    {

        printf("%s\t\tL-->,SL' \n",ip);

        ip++;

        if(S())

        {

            if (Ldash())

            {

                return 1;

            }

            else

            {

                return 0;

            }

        }

        else

        {

            return 0;

        }

    }

    else

    {

        printf("%s\t\tL'-->e \n",ip);

        return 1;

    }

}

int S()

{

    if(\*ip =='(')

    {

        printf("%s\t\tS-->(L) \n",ip);

        ip++;

        if(L())

        {

            if (\*ip ==')')

            {

                ip++;

                return 1;

            }

            else

            {

                return 0;

            }

        }

        else

        {

            return 0;

        }

    }

    else if (\*ip == 'a')

    {

        ip++;

        printf("%s\t\tS-->a \n",ip);

        return 1;

    }

    else

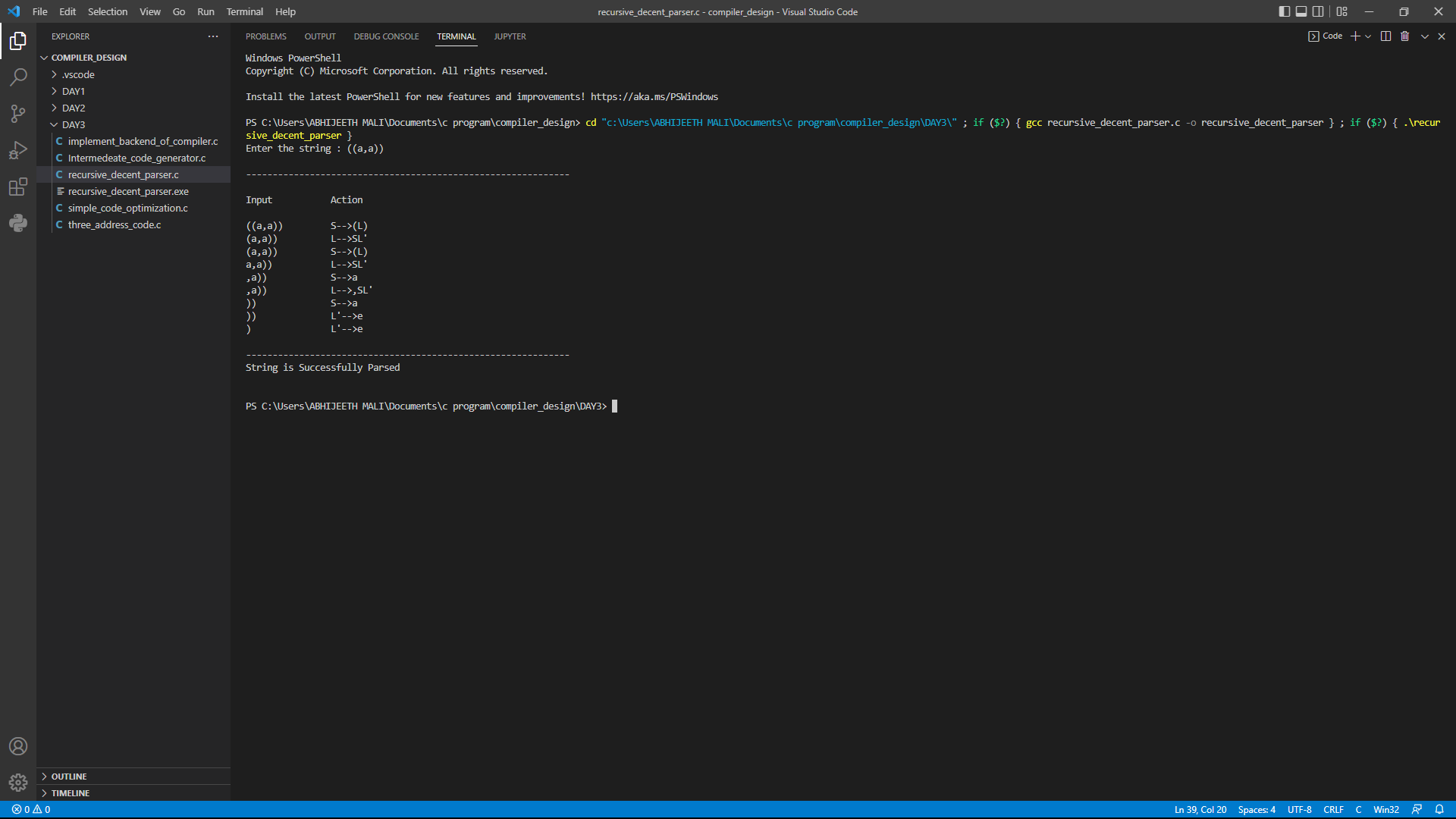
    {

        return 0;

    }

}

Output :



4 simple\_code\_optimization

Source code :

#include <stdio.h>

int main(void)

{

    int fact[5];

    fact[0] = 1;

    for (int i = 1; i <= 5; ++i)

    {

        fact[i] = fact[i - 1] \* i;

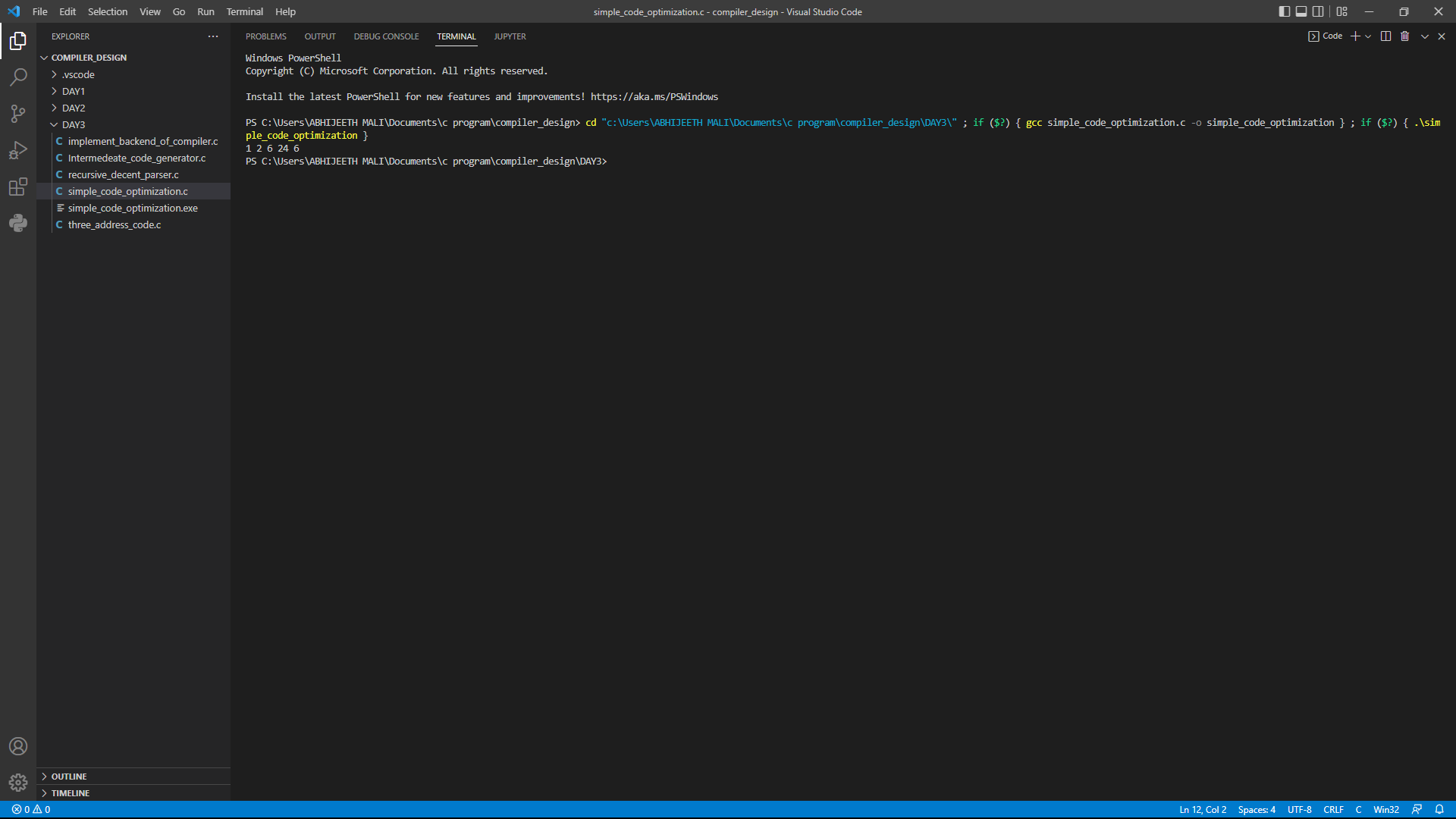
        printf("%d ", fact[i]);

    }

    return 0;

}

Output :



5. three\_address\_code

Source code :

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int pm();

int plus();

int division();

int i, ch, j, l, addr = 100;

char a[10], b[10], c[10], d[10], id1[5], op[5], id2[5];

int main()

{

    // clrscr();

    while (1)

    {

        printf("\n1.assignment\n2.arithmetic\n3.relational\n4.Exit\nEnter the choice:");

        scanf("%d", &ch);

        switch (ch)

        {

        case 1:

            printf("\nEnter the expression with assignment operator:");

            scanf("%s", b);

            l = strlen(b);

            d[0] = '\0';

            i = 0;

            while (b[i] != '=')

            {

                i++;

            }

            strncat(d, b, i);

            strrev(b);

            c[0] = '\0';

            strncat(c, b, l - (i + 1));

            strrev(c);

            printf("Three address code:\ntemp=%s\n%s=temp\n", c, d);

            break;

        case 2:

            printf("\nEnter the expression with arithmetic operator:");

            scanf("%s", a);

            strcpy(b, a);

            l = strlen(b);

            c[0] = '\0';

            for (i = 0; i < l; i++)

            {

                if (b[i] == '+' || b[i] == '-')

                {

                    if (b[i + 2] == '/' || b[i + 2] == '\*')

                    {

                        pm();

                        break;

                    }

                    else

                    {

                        plus();

                        break;

                    }

                }

                else if (b[i] == '/' || b[i] == '\*')

                {

                    division

                ();

                    break;

                }

            }

            break;

        case 3:

            printf("Enter the expression with relational operator");

            scanf("%s%s%s", &id1, &op, &id2);

            if (((strcmp(op, "<") == 0) || (strcmp(op, ">") == 0) || (strcmp(op, "<=") == 0) || (strcmp(op, ">=") == 0) || (strcmp(op, "==") == 0) || (strcmp(op, "!=") == 0)) == 0)

                printf("Expression is error");

            else

            {

                printf("\n%d\tif %s%s%s goto %d", addr, id1, op, id2, addr + 3);

                addr++;

                printf("\n%d\t T:=0", addr);

                addr++;

                printf("\n%d\t goto %d", addr, addr + 2);

                addr++;

                printf("\n%d\t T:=1", addr);

            }

            break;

        case 4:

            exit(0);

        }

    }

}

int pm()

{

    strrev(b);

    j = l - i - 1;

    strncat(c, b, j);

    strrev(c);

    printf("Three address code:\ntemp=%s\ntemp1=%c%ctemp\n", c, b[j + 1], b[j]);

}

int division()

{

    strncat(c, b, i + 2);

    printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n", c, b[i + 2], b[i + 3]);

}

int plus()

{

    strncat(c, b, i + 2);

    printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n", c, b[i + 2], b[i + 3]);

}

Out put :

