

CSE 2003

DATA STRUCTURES AND ALGORITHMS



Lab Assessment – 2

L19+L20 | SJT317

FALL SEMESTER 2020-21

by

SHARADINDU ADHIKARI

19BCE2105

Question 1

Problem:

Write a program for converting the given decimal number into binary.

Code:

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

long decimalToBinary(int decimalnum)
{
    long binarynum = 0;
    int rem, temp = 1;

    while (decimalnum!=0)
    {
        rem = decimalnum%2;
        decimalnum = decimalnum / 2;
        binarynum = binarynum + rem*temp;
        temp = temp * 10;
    }
    return binarynum;
}

int main()
{
    int decimalnum;
    printf("Enter a Decimal Number: ");
    scanf("%d", &decimalnum);
    printf("Equivalent Binary Number is: %ld",
decimalToBinary(decimalnum));
    return 0;
}
```

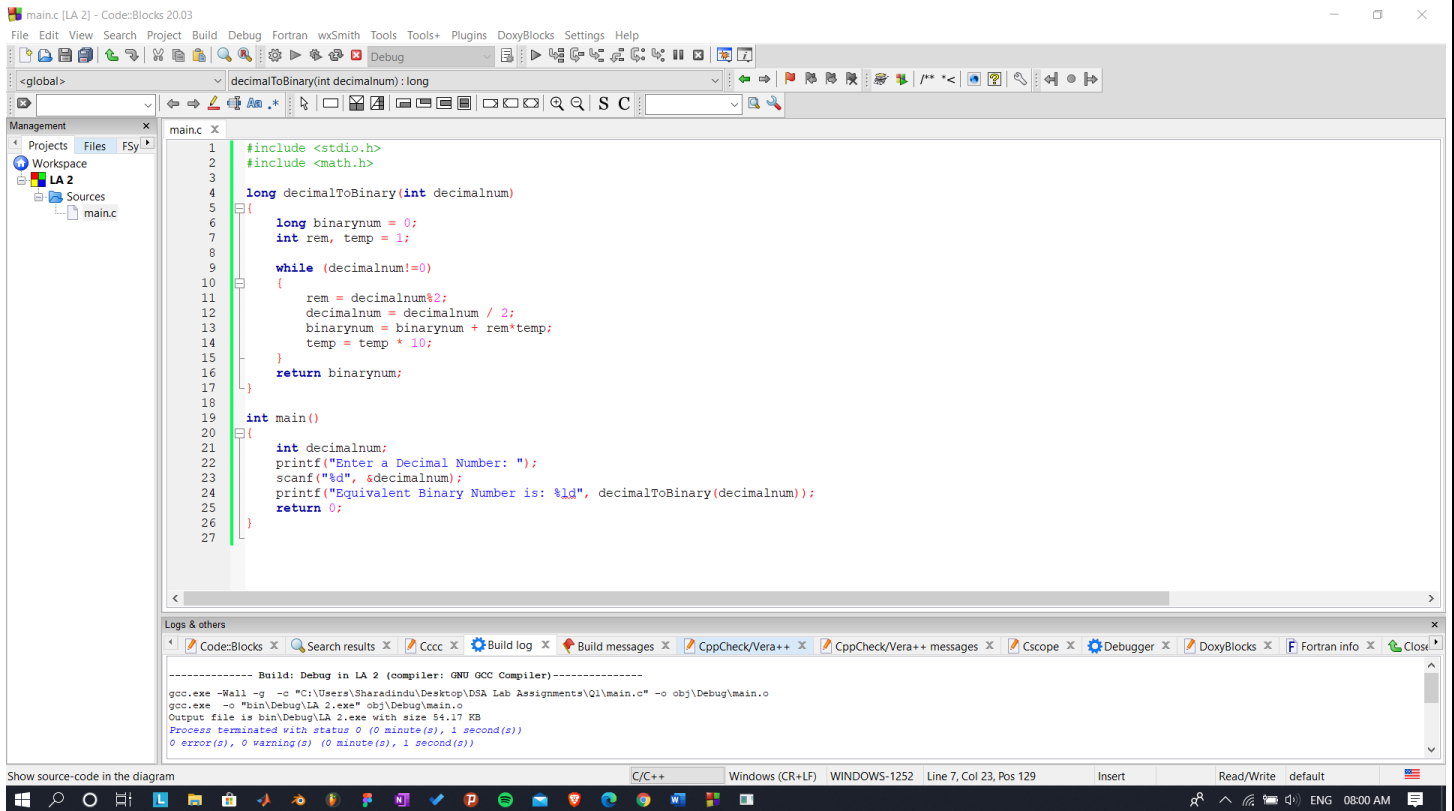
Input:

Enter a Decimal Number: 210

Output:

Equivalent Binary Number is: 11010010

Screenshots of Code & CMD:



```
main.c [LA 2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> decimalToBinary(int decimalnum): long
Management
Projects Files FSy
Workspace
LA 2
Sources
main.c
main.c X
1 #include <stdio.h>
2 #include <math.h>
3
4 long decimalToBinary(int decimalnum)
5 {
6     long binarynum = 0;
7     int rem, temp = 1;
8
9     while (decimalnum!=0)
10    {
11        rem = decimalnum%2;
12        decimalnum = decimalnum / 2;
13        binarynum = binarynum + rem*temp;
14        temp = temp * 10;
15    }
16    return binarynum;
17 }
18
19 int main()
20 {
21     int decimalnum;
22     printf("Enter a Decimal Number: ");
23     scanf("%d", &decimalnum);
24     printf("Equivalent Binary Number is: %ld", decimalToBinary(decimalnum));
25     return 0;
26 }
27
Logs & others
Code::Blocks X Search results X Cccc X Build log X Build messages X CppCheck/Vera++ X CppCheck/Vera++ messages X Cscope X Debugger X DoxyBlocks X Fortran info X Close
----- Build: Debug in LA 2 (compiler: GNU GCC Compiler)-----
gcc.exe -Wall -g -c "C:\Users\Sharadindu\Desktop\DSA Lab Assignments\Q1\main.c" -o obj\Debug\main.o
gcc.exe -o "bin\Debug\LA 2.exe" obj\Debug\main.o
Output file is bin\Debug\LA 2.exe with size 54.17 KB
Process terminated with status 0 (0 minute(s), 1 second(s))
0 error(s), 0 warning(s) (0 minute(s), 1 second(s))
Show source-code in the diagram C/C++ Windows (CR+LF) WINDOWS-1252 Line 7, Col 23, Pos 129 Insert Read/Write default ENG 08:00 AM
```

```
"C:\Users\Sharadindu\Desktop\DSA Lab Assignments\Q1\bin\Debug\LA 2.exe"
Enter a Decimal Number: 210
Equivalent Binary Number is: 11010010
Process returned 0 (0x0)   execution time : 1.915 s
Press any key to continue.
```

Question 2

Problem:

Write a program to implement a stack in an array and perform PUSH, POP, PEEP and CHANGE operations on it using functions.

Code & Input:

```
#include<stdio.h>
#define size 5
struct stack{
    int a[size],top;
    int temp[size], tos;
}s;
// Push operation....
void push(int item){
    s.a[++s.top] = item;
}
// Pop operation....
int pop(){
    return s.a[s.top--];
}
// Display operation....
void display(){
    int i;
    printf("\nThe stack contains: ");
    for(i = s.top; i>=0; i--){
        printf("\n\t%d", s.a[i]);
    }
}
// Peep operation....
void peep(){
    printf("\n\tTop : %d", s.top);
    printf("\n\tValue: %d",s.a[s.top]);
}
void change(int row, int new_element){
    int i;
    int j = -1;
    printf("\n\tTop: %d", s.top);
    for(i=s.top; i>row; i--){
        s.temp[++j] = s.a[s.top--];
    }
    s.a[s.top] = new_element;

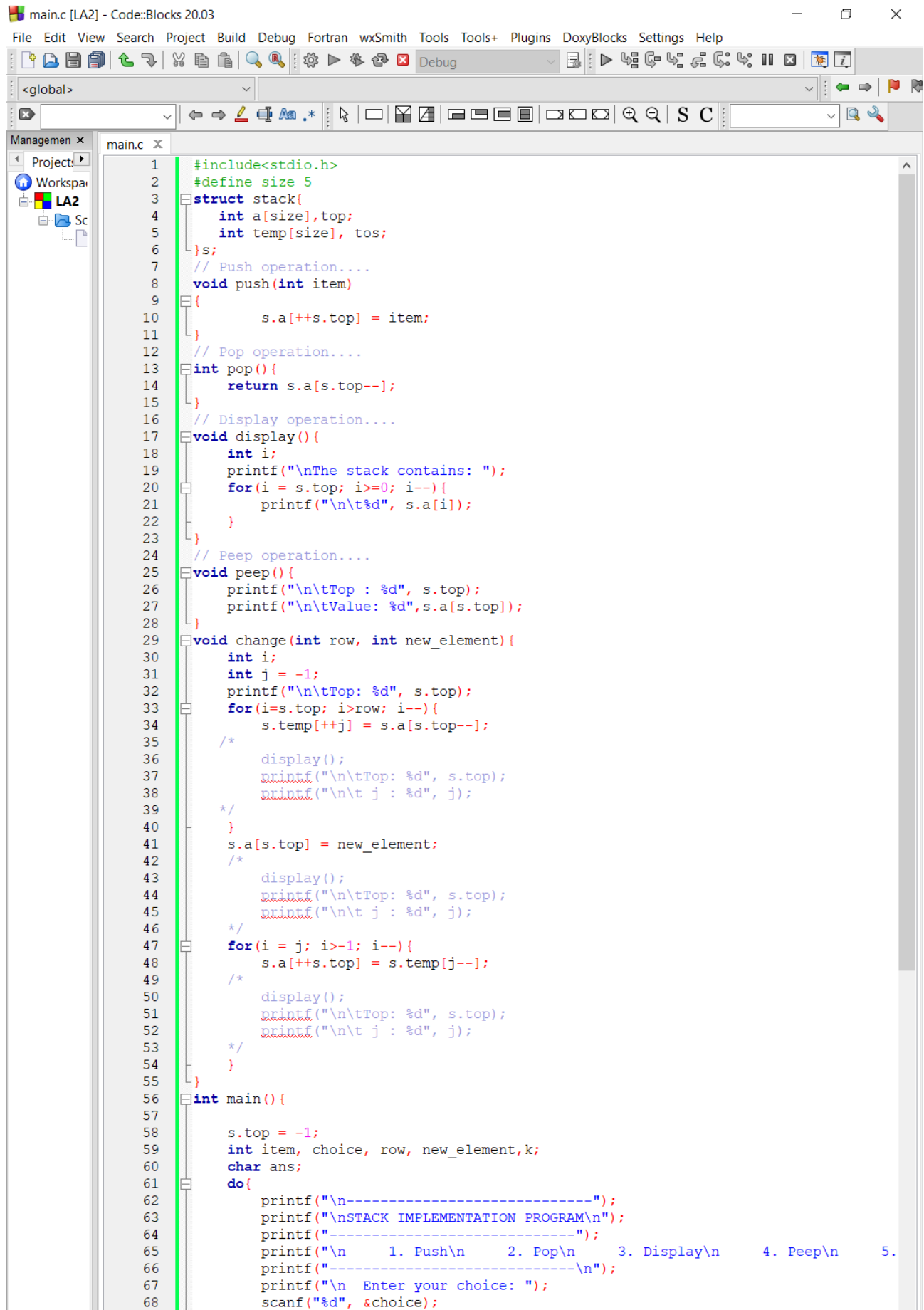
    for(i = j; i>=-1; i--){
        s.a[++s.top] = s.temp[j--];
    }
}
```

```

    }
}
int main(){
    s.top = -1;
    int item, choice, row, new_element;
    char ans;
    do{
        printf("\n-----");
        printf("\nSTACK IMPLEMENTATION PROGRAM\n");
        printf("-----");
        printf("\n      1. Push\n      2. Pop\n      3. Display\n      4.
Peep\n      5. Change\n      6. Exit\n");
        printf("-----\n");
        printf("\n  Enter your choice: ");
        scanf("%d", &choice);
        switch(choice){
            case 1:
                if(s.top >= size-1){
                    printf("\nStack overflow..\n");
                    break;
                }
                printf("\nEnter item to be pushed: ");
                scanf("%d", &item);
                push(item);
                break;
            case 2:
                if(s.top == -1){
                    printf("\n..Stack underflow..\n");
                    break;
                }
                pop();
                break;
            case 3:
                display();
                break;
            case 4:
                peep();
                break;
            case 5:
                printf("\n\tEnter row no : ");
                scanf("%d",&row);
                printf("\n\tEnter new element: ");
                scanf("%d", &new_element);
                change(row, new_element );
                break;
            case 6:
                return 0;
        }
    }while(choice != 6);
    return 0;
}

```

Screenshot of Code & Output:



```
main.c [LA2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
main.c
1 #include<stdio.h>
2 #define size 5
3 struct stack{
4     int a[size],top;
5     int temp[size], tos;
6 }s;
7 // Push operation...
8 void push(int item)
9 {
10     s.a[++s.top] = item;
11 }
12 // Pop operation...
13 int pop(){
14     return s.a[s.top--];
15 }
16 // Display operation...
17 void display(){
18     int i;
19     printf("\nThe stack contains: ");
20     for(i = s.top; i>=0; i--){
21         printf("\n\t%d", s.a[i]);
22     }
23 }
24 // Peep operation...
25 void peep(){
26     printf("\n\tTop : %d", s.top);
27     printf("\n\tValue: %d",s.a[s.top]);
28 }
29 void change(int row, int new_element){
30     int i;
31     int j = -1;
32     printf("\n\tTop: %d", s.top);
33     for(i=s.top; i>row; i--){
34         s.temp[++j] = s.a[s.top--];
35     /*
36         display();
37         printf("\n\tTop: %d", s.top);
38         printf("\n\t j : %d", j);
39     */
40     }
41     s.a[s.top] = new_element;
42     /*
43         display();
44         printf("\n\tTop: %d", s.top);
45         printf("\n\t j : %d", j);
46     */
47     for(i = j; i>=-1; i--){
48         s.a[++s.top] = s.temp[j--];
49     /*
50         display();
51         printf("\n\tTop: %d", s.top);
52         printf("\n\t j : %d", j);
53     */
54     }
55 }
56 int main(){
57     s.top = -1;
58     int item, choice, row, new_element,k;
59     char ans;
60     do{
61         printf("\n-----");
62         printf("\nSTACK IMPLEMENTATION PROGRAM\n");
63         printf("-----");
64         printf("\n    1. Push    2. Pop    3. Display    4. Peep    5.
65         printf("-----\n");
66         printf("\n Enter your choice: ");
67         scanf("%d", &choice);
68     }
```

```
69         switch(choice){
70         case 1:
71             if(s.top >= size-1){
72                 printf("\nStack overflow..\n");
73                 break;
74             }
75             printf("\nEnter the number of items to be pushed:");
76             scanf("%d",&k);
77             printf("\nEnter items to be pushed: ");
78             for(int i=0;i<k;i++){
79                 {
80                     scanf("%d",&item);
81                     push(item);
82                 }
83                 break;
84             case 2:
85                 if(s.top == -1)
86                 {
87                     printf("\n..Stack underflow..\n");
88                     break;
89                 }
90                 pop();
91                 break;
92             case 3:
93                 display();
94                 break;
95             case 4:
96                 peep();
97                 break;
98             case 5:
99                 printf("\n\tEnter row no : ");
100                 scanf("%d",&row);
101                 printf("\n\tEnter new element: ");
102                 scanf("%d", &new_element);
103                 change(row, new_element );
104                 break;
105             case 6:
106                 return 0;
107             }
108         }while(choice != 6);
109         return 0;
110     }
111 }
```

Logs & others

File	Line	Message
=== Build: Debug in LA2 (compiler: GNU GCC Compiler) ===		
In function 'main':		
C:\Users\Sh...	60	warning: unused variable 'ans' [-Wunused-variable]
=== Build finished: 0 error(s), 1 warning(s) (0 minute(s), 2 second(s)) ===		

C/C++ Windows (CR+LF) WINDOWS-1252 Line 111, Col 1, Pos 2740 Insert Read/Write default ENG 05:16 PM

```
"C:\Users\Sharadindu\Desktop\DSA Lab Assignments\LA2\bin\Debug\LA2.exe"

-----
STACK IMPLEMENTATION PROGRAM
-----
1. Push
2. Pop
3. Display
4. Peep
5. Change
6. Exit
-----

Enter your choice: 1

Enter the number of items to be pushed:5

Enter items to be pushed: 1
2
3
4
5
```

STACK IMPLEMENTATION PROGRAM

1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-

Enter your choice: 2

STACK IMPLEMENTATION PROGRAM

1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-

Enter your choice: 3

The stack contains:

4
3
2
1

STACK IMPLEMENTATION PROGRAM

1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-

Enter your choice: 4

Top : 3
Value: 4

STACK IMPLEMENTATION PROGRAM

1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-

Enter your choice: 5

Enter row no : 3

Enter new element: 7

Top: 3

STACK IMPLEMENTATION PROGRAM

1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-

Enter your choice: 6

Process returned 0 (0x0) execution time : 48.261 s
Press any key to continue.

Question 3

Problem:

WAP to convert the following expression to its postfix equivalent using stack:

- $((A + B) * D) ^ (E - F)$
- $A + (B * C - (D / E ^ F) * G) * H$, where $^$ denotes raise to the power.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#define SIZE 100

char stack[SIZE];
int top = -1;
void push(char item)
{
    if(top >= SIZE-1)
    {
        printf("\nStack Overflow.");
    }
    else
    {
        top = top+1;
        stack[top] = item;
    }
}
char pop()
{
    char item ;
    if(top <0)
    {
        printf("stack under flow: invalid infix expression");
        getchar();
        /* underflow may occur for invalid expression */
        /* where ( and ) are not matched */
        exit(1);
    }
    else
    {
        item = stack[top];
        top = top-1;
        return(item);
    }
}
int is_operator(char symbol)
{
    if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+'
    || symbol == '-')
    {

```

```

        return 1;
    }
    else
    {
        return 0;
    }
}
int precedence(char symbol)
{
    if(symbol == '^')
    {
        return(3);
    }
    else if(symbol == '*' || symbol == '/')
    {
        return(2);
    }
    else if(symbol == '+' || symbol == '-')
    {
        return(1);
    }
    else
    {
        return(0);
    }
}
void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
    int i, j;
    char item;
    char x;
    push('(');
    strcat(infix_exp, " ");
    i=0;
    j=0;
    item=infix_exp[i];
    while(item != '\0')
    {
        if(item == '(')
        {
            push(item);
        }
        else if( isdigit(item) || isalpha(item))
        {
            postfix_exp[j] = item;
            j++;
        }
        else if(is_operator(item) == 1)
        {
            x=pop();
            while(is_operator(x) == 1 && precedence(x) >= precedence(item))
            {
                postfix_exp[j] = x;
                j++;
                x = pop();
            }

```

```

        push(x);
        push(item);
    }
    else if(item == ')')
    {
        x = pop();
        while(x != '(')
        {
            postfix_exp[j] = x;
            j++;
            x = pop();
        }
    }
    else
    {
        printf("\nInvalid infix Expression.\n");
        getchar();
        exit(1);
    }
    i++;
    item = infix_exp[i];
}
if(top>0)
{
    printf("\nInvalid infix Expression.\n");
    getchar();
    exit(1);
}
if(top>0)
{
    printf("\nInvalid infix Expression.\n");
    getchar();
    exit(1);
}
postfix_exp[j] = '\0';
}
int main()
{
    char infix[SIZE], postfix[SIZE];
    printf("ASSUMPTION: The infix expression contains single letter
variables and single digit constants only.\n");
    printf("\nEnter Infix expression : ");
    gets(infix);
    InfixToPostfix(infix,postfix);
    printf("Postfix Expression: ");
    puts(postfix);
    return 0;
}

```

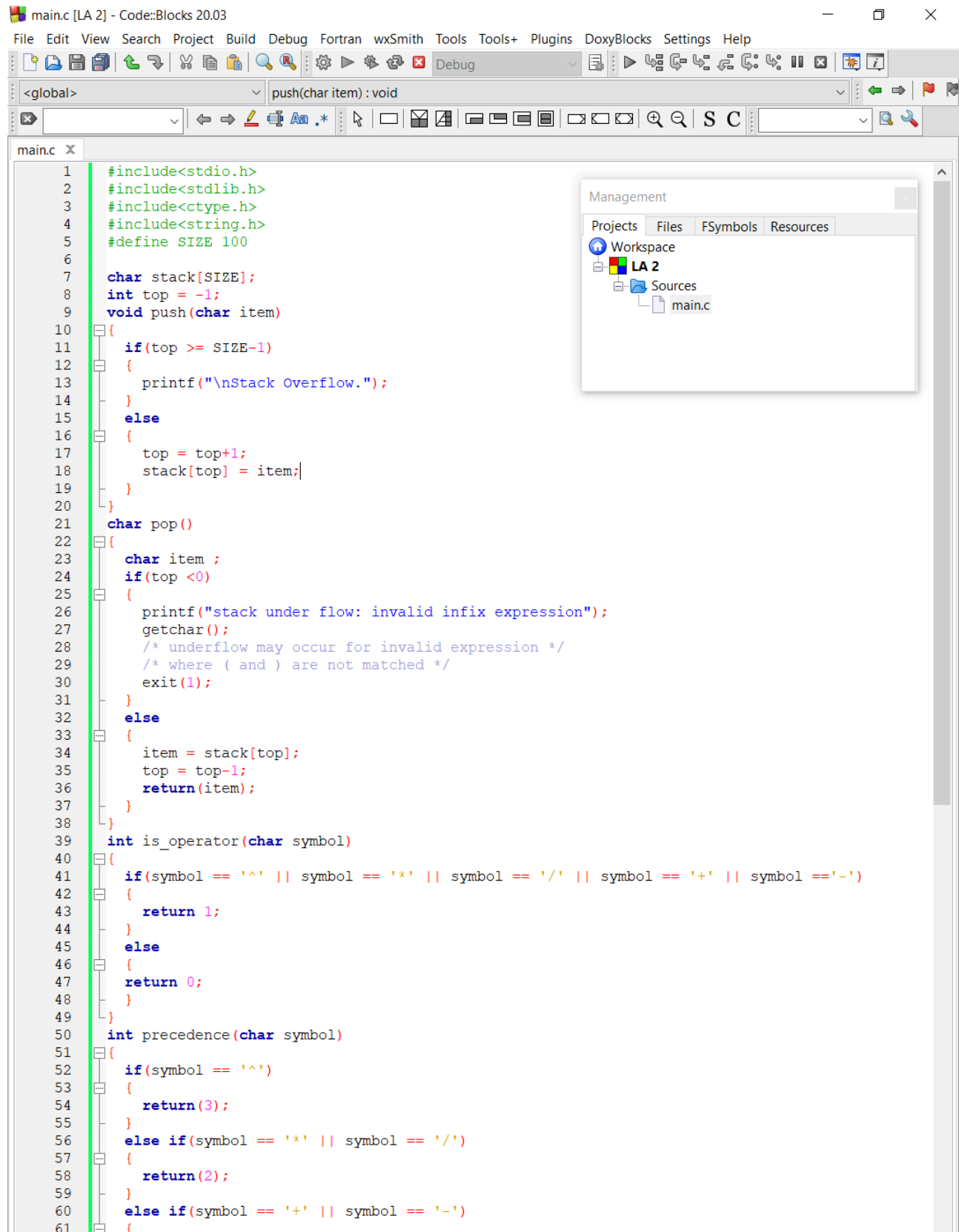
Inputs:

- a. Enter Infix expression: $((A+B)*D)^{(E-F)}$
- b. Enter Infix expression: $A+(B*C-(D/E^F)*G)*H$

Outputs:

- a. Postfix Expression: $AB+D*EF-^$
- b. Postfix Expression: $ABC*DEF^/G*-H*+$

Screenshot of Code & CMD:



```
main.c [LA 2] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global> push(char item): void
main.c X
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<ctype.h>
4 #include<string.h>
5 #define SIZE 100
6
7 char stack[SIZE];
8 int top = -1;
9 void push(char item)
10 {
11     if(top >= SIZE-1)
12     {
13         printf("\nStack Overflow.");
14     }
15     else
16     {
17         top = top+1;
18         stack[top] = item;
19     }
20 }
21 char pop()
22 {
23     char item ;
24     if(top < 0)
25     {
26         printf("stack under flow: invalid infix expression");
27         getchar();
28         /* underflow may occur for invalid expression */
29         /* where ( and ) are not matched */
30         exit(1);
31     }
32     else
33     {
34         item = stack[top];
35         top = top-1;
36         return(item);
37     }
38 }
39 int is_operator(char symbol)
40 {
41     if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-')
42     {
43         return 1;
44     }
45     else
46     {
47         return 0;
48     }
49 }
50 int precedence(char symbol)
51 {
52     if(symbol == '^')
53     {
54         return(3);
55     }
56     else if(symbol == '*' || symbol == '/')
57     {
58         return(2);
59     }
60     else if(symbol == '+' || symbol == '-')
61     {
62         return(1);
63     }
64 }
```

```

62     return(1);
63 }
64 else
65 {
66     return(0);
67 }
68 }
69 void InfixToPostfix(char infix_exp[], char postfix_exp[])
70 {
71     int i, j;
72     char item;
73     char x;
74     push('(');
75     strcat(infix_exp, " ");
76     i=0;
77     j=0;
78     item=infix_exp[i];
79     while(item != '\0')
80     {
81         if(item == '(')
82         {
83             push(item);
84         }
85         else if( isdigit(item) || isalpha(item))
86         {
87             postfix_exp[j] = item;
88             j++;
89         }
90         else if(is_operator(item) == 1)
91         {
92             x=pop();
93             while(is_operator(x) == 1 && precedence(x)>= precedence(item))
94             {
95                 postfix_exp[j] = x;
96                 j++;
97                 x = pop();
98             }
99             push(x);
100             push(item);
101         }
102         else if(item == ')')
103         {
104             x = pop();
105             while(x != '(')
106             {
107                 postfix_exp[j] = x;
108                 j++;
109                 x = pop();
110             }
111         }
112         else
113         {
114             printf("\nInvalid infix Expression.\n");
115             getchar();
116             exit(1);
117         }
118         i++;
119         item = infix_exp[i];
120     }
121     if(top>0)
122     {
123         printf("\nInvalid infix Expression.\n");
124         getchar();
125         exit(1);
126     }
127     if(top>0)
128     {
129         printf("\nInvalid infix Expression.\n");
130         getchar();
131         exit(1);
132     }
133     postfix_exp[j] = '\0';
134 }
135
136 int main()
137 {
138     char infix[SIZE], postfix[SIZE];
139     printf("ASSUMPTION: The infix expression contains single letter variables and single digit c
140     printf("\nEnter Infix expression : ");
141     gets(infix);
142     InfixToPostfix(infix,postfix);

```

```
143     printf("Postfix Expression: ");
144     puts(postfix);
145     return 0;
146 }
147
```

Logs & others

Code::Blocks x Search results x Cccc x Build log x Build messages x CppCheck/Vera++ x CppCheck/Vera++ mess

set variable: PATH=.;C:\Program Files\CodeBlocks\mingw\bin;C:\Program Files\CodeBlocks\mingw\bin\..\bin;C:\Program Files (x86)\Common Files\Intel\Shared Libraries\redist\intel64\compiler;C:\Windows\System32;C:\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\OrCAD\OrCAD_16.6_Lite\tools\pspice;C:\OrCAD\OrCAD_16.6_Lite\tools\capture;C:\OrCAD\OrCAD_16.6_Lite\tools\bin;C:\OrCAD\OrCAD_16.6_Lite\OpenAccess\bin\Win32\opt;C:\OrCAD\OrCAD_16.6_Lite\tools\fet\bin;C:\OrCAD\OrCAD_16.6_Lite\tools\pcb\bin;C:\Program Files\MiKTeX 2.9\miktex\bin\x64;C:\Program Files\Git\cmd;C:\Program Files\nodejs;C:\Program Files\MATLAB\R2020a\runtime\win64;C:\Program Files\MATLAB\R2020a\bin;C:\Users\Sharadindu\AppData\Local\Microsoft\WindowsApps;C:\Users\Sharadindu\AppData\Local\Programs\Microsoft VS Code

S/C/C++ Windows (CR+LF) WINDOWS-1252 Line 18, Col 23, Pos 275 Insert Read/Write default ENG 09:22 AM

```
"C:\Users\Sharadindu\Desktop\DSA Lab Assignments\Q3\bin\Debug\LA 2.exe"
ASSUMPTION: The infix expression contains single letter variables and single digit constants only.
Enter Infix expression : ((A+B)*D)^(E-F)
Postfix Expression: AB+D*EF-^
Process returned 0 (0x0)   execution time : 24.164 s
Press any key to continue.
```

```
"C:\Users\Sharadindu\Desktop\DSA Lab Assignments\Q3\bin\Debug\LA 2.exe"
ASSUMPTION: The infix expression contains single letter variables and single digit constants only.
Enter Infix expression : A+(B*C-(D/E^F)*G)*H
Postfix Expression: ABC*DEF^/G*-H*+
Process returned 0 (0x0)   execution time : 43.765 s
Press any key to continue.
```

Question 4

Problem:

Implement a stack S of n elements using arrays. Write functions to perform PUSH and POP operations. Implement queries using push and pop functions to

- retrieve the m^{th} element of the stack S from the top ($m < n$), leaving the stack without its top $m - 1$ elements.
- retain only the elements in the odd position of the stack and pop out all even positioned elements.

Code & Input:

```
#include<stdio.h>
#define max 5 //max size of the Stack
int top,a[max];
void push(void) //insert element on top
{
    int x,i;
    if(top==max-1) // Condition for checking If Stack is Full
    {
        printf("stack overflow\n");
        return;
    }
    printf("Enter Element to be inserted into Stack\n");
    scanf("%d",&x);
    a[++top]=x; //increment the top and inserting element
    printf("Element inserted into Stack\n");
    return;
}
void pop(void) //function to remove one element from top
{
    if(top==-1) // Condition for checking If Stack is Empty
    {
        printf("stack underflow\n");
        return;
    }
    a[top--]=0; //insert 0 at place of removing element and decrement the top
    return;
}
void display(void) //function to display elems in stack
{
    int i;
    if(top==-1)
    {
        printf("\n Stack is empty\n");
        return;
    }
    printf("\n Elements of Stack are :\n");
    for(i=0;i<=top;i++)
    {
        printf("%d\n",a[i]);
    }
}
```

```

}
return;
}
void popeven(void) //function to pop even position and retrieve odd
positions
{
int i,t=0,temp[max],temptop=0;
if(top==-1) //checking if stack is empty
{
printf("\n Stack is empty ! \n");
return;
}
for(i=0;i<=top;i++) //checking odd position elements from stack
{
if((i+1)%2 != 0)
{
temp[t]=a[i];
t++;
}
}
temptop=top; //temporary variable to store value of top
for(i=0;i<=temptop;i++) //forloop to remove all elemnts from stack
{
pop();
}
for(i=0;i<=t;i++) //forloop to push only odd position elemnts back into
stack
{
top++;
a[i]=temp[i];
}
printf("Elements of New Stack are :\n");
top--;
for(i=0;i<=top;i++) //display new stack
{
printf("%d\n",a[i]);
}
return;
}
void main(void) //main function
{
int c; top=-1;
system("cls");
do
{
printf("\n STACK OPERATIONS\n");
printf("\n 1:PUSH\n 2:POP from top(Mth element)\n 3:POP Even
positions\n 4:Display\n 5:Exit\n Choice:");
scanf("%d",&c);
switch(c)
{
case 1:push();printf("Maximum 5 elements !");
break;
case 2:pop();printf("Mth element popped! \n");
break;
case 3:popeven();

```



```

break;
case 4:display();
break;
case 5:printf("Program Ends\n");
break;
default :printf("Wrong Choice ! Enter Again !\n");
break;
}
}while(c!=5);
}

```

Screenshot of Code & Output:

The screenshot shows the Code::Blocks IDE with the following code in main.c:

```

1  #include<stdio.h>
2  #define max 5 //max size of the Stack
3  int top,a[max];
4  void push(void) //insert element on top
5  {
6      int x,i;
7      if(top==max-1) // Condition for checking If Stack is Full
8      {
9          printf("stack overflow\n");
10         return;
11     }
12     printf("Enter Element to be inserted into Stack\n");
13     scanf("%d",&x);
14     a[++top]=x; //increment the top and inserting element
15     printf("Element inserted into Stack\n");
16     return;
17 }
18 void pop(void) //function to remove one element from top
19 {
20     if(top== -1) // Condition for checking If Stack is Empty
21     {
22         printf("stack underflow\n");
23         return;
24     }
25     a[top--]=0; //insert 0 at place of removing element and decrement the top
26     return;
27 }
28 void display(void) //function to display elems in stack
29 {
30     int i;
31     if(top== -1)
32     {
33         printf("\n Stack is empty\n");
34         return;
35     }
36     printf("\n Elements of Stack are :\n");
37     for(i=0;i<=top;i++)
38     {
39         printf("%d\n",a[i]);
40     }
41     return;
42 }
43 void popeven(void) //function to pop even position and retrieve odd positions
44 {
45     int i,t=0,temp[max],temptop=0;
46     if(top== -1) //checking if stack is empty
47     {
48         printf("\n Stack is empty ! \n");
49         return;
50     }
51     for(i=0;i<=top;i++) //checking odd position elements from stack
52     {

```

```

53     if((i+1)%2 != 0)
54     {
55         temp[t]=a[i];
56         t++;
57     }
58     }
59     temptop=top; //temporary variable to store value of top
60     for(i=0;i<=temptop;i++) //forloop to remove all elemnts from stack
61     {
62         pop();
63     }
64     for(i=0;i<=t;i++) //forloop to push only odd position elemnts back into stack
65     {
66         top++;
67         a[i]=temp[i];
68     }
69     printf("Elements of New Stack are :\n");
70     top--;
71     for(i=0;i<=top;i++) //display new stack
72     {
73         printf("%d\n",a[i]);
74     }
75     return;
76 }
77 void main(void) //main function
78 {
79     int c; top=-1;
80     system("cls");
81     do
82     {
83         printf("\n STACK OPERATIONS\n");
84         printf("\n 1:PUSH\n 2:POP from top(Mth element)\n 3:POP Even positions\n 4:Display\n");
85         scanf("%d",&c);
86         switch(c)
87         {
88             case 1:push();printf("Maximum 5 elements !");
89             break;
90             case 2:pop();printf("Mth element popped! \n");
91             break;
92             case 3:popeven();
93             break;
94             case 4:display();
95             break;
96             case 5:printf("Program Ends\n");
97             break;
98             default :printf("Wrong Choice ! Enter Again !\n");
99             break;
100         }
101     }while(c!=5);
102 }
103

```

Logs & others

Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppChe

Set variable: PATH=C:\Program Files\Codeblocks\mingw\bin;C:\Program Files\Codeblocks\mingw\bin;C:\Program Files (x86)\Common Files\Intel\Shared Libraries\redist\intel64\compiler;C:\Windows\System32;C:\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\OrCAD\OrCAD_16.6_Lite\tools\pspice;C:\OrCAD\OrCAD_16.6_Lite\tools\capture;C:\OrCAD\OrCAD_16.6_Lite\tools\bin;C:\OrCAD\OrCAD_16.6_Lite\OpenAccess\bin\Win32\opt;C:\OrCAD\OrCAD_16.6_Lite\tools\fet\bin;C:\OrCAD\OrCAD_16.6_Lite\tools\pcb\bin;C:\Program Files\MiKTeX 2.9\miktex\bin\x64;C:\Program Files\Git\cmd;C:\Program Files\nodejs;C:\Program Files\MATLAB\R2020a\runtime\win64;C:\Program Files\MATLAB\R2020a\bin;C:\Users

"C:\Users\Sharadindu\Desktop\DSA Lab Assignments\LA2\bin\Debug\LA2.exe"

STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)
3:POP Even positions
4:Display
5:Exit

Choice:1
Enter Element to be inserted into Stack
3

Element inserted into Stack
Maximum 5 elements !
STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)

```
3:POP Even positions
4:Display
5:Exit
Choice:1
Enter Element to be inserted into Stack
7
Element inserted into Stack
Maximum 5 elements !
STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)
3:POP Even positions
4:Display
5:Exit
Choice:1
Enter Element to be inserted into Stack
9
Element inserted into Stack
Maximum 5 elements !
STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)
3:POP Even positions
4:Display
5:Exit
Choice:1
Enter Element to be inserted into Stack
10
Element inserted into Stack
Maximum 5 elements !
STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)
3:POP Even positions
4:Display
5:Exit
Choice:4

Elements of Stack are :
3
7
9
10

STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)
3:POP Even positions
4:Display
5:Exit
Choice:3
Elements of New Stack are :
3
9

STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)
3:POP Even positions
4:Display
5:Exit
Choice:2
Mth element popped!

STACK OPERATIONS

1:PUSH
2:POP from top(Mth element)
3:POP Even positions
4:Display
5:Exit
Choice:4

Elements of Stack are :
3
```

Question 5

Problem:

Implement a program to evaluate any given postfix expression. Test your program for the evaluation of the equivalent postfix form of the expression:

$(-(A*B)/D) \uparrow C + E - F * H * I$ for $A = 1, B = 2, D = 3, C = 14, E = 110, F = 220, H = 16.78, I = 364.621$.

Code & Input:

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

// Stack type
struct Stack
{
    int top;
    unsigned capacity;
    int* array;
};

// Stack Operations
struct Stack* createStack( unsigned capacity )
{
    struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack));

    if (!stack) return NULL;

    stack->top = -1;
    stack->capacity = capacity;
    stack->array = (int*) malloc(stack->capacity * sizeof(int));

    if (!stack->array) return NULL;

    return stack;
}

int isEmpty(struct Stack* stack)
{
    return stack->top == -1 ;
}

int peek(struct Stack* stack)
{
    return stack->array[stack->top];
}

int pop(struct Stack* stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--] ;
}
```

```

        return '$';
    }

void push(struct Stack* stack,int op)
{
    stack->array[++stack->top] = op;
}

// The main function that returns value
// of a given postfix expression
int evaluatePostfix(char* exp)
{
    // Create a stack of capacity equal to expression size
    struct Stack* stack = createStack(strlen(exp));
    int i;

    // See if stack was created successfully
    if (!stack) return -1;

    // Scan all characters one by one
    for (i = 0; exp[i]; ++i)
    {
        //if the character is blank space then continue
        if(exp[i]==' ')continue;

        // If the scanned character is an
        // operand (number here),extract the full number
        // Push it to the stack.
        else if (isdigit(exp[i]))
        {
            int num=0;

            //extract full number
            while(isdigit(exp[i]))
            {
                num=num*10 + (int)(exp[i]-'0');
                i++;
            }
            i--;

            //push the element in the stack
            push(stack,num);
        }

        // If the scanned character is an operator, pop two
        // elements from stack apply the operator
        else
        {
            int val1 = pop(stack);
            int val2 = pop(stack);

            switch (exp[i])
            {
                {
                    case '+': push(stack, val2 + val1); break;
                    case '-': push(stack, val2 - val1); break;

```

```

        case '*': push(stack, val2 * val1); break;
        case '/': push(stack, val2/val1); break;

    }

}

return pop(stack);
}

// Driver program to test above functions
int main()
{
    char exp[] = "(-(1*2)/3)^14+110-220*16.78*346.621";
    printf ("%d", evaluatePostfix(exp));
    return 0;
}

```

Screenshot of Code & Output:

The screenshot shows a code editor window titled '*main.c [LA2] - Code::Blocks 20.03'. The editor displays the following C code:

```

1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4  #include <stdlib.h>
5
6  struct Stack
7  {
8      int top;
9      unsigned capacity;
10     int* array;
11 };
12
13 struct Stack* createStack( unsigned capacity )
14 {
15     struct Stack* stack = (struct Stack*) malloc(sizeof(struct Stack));
16
17     if (!stack) return NULL;
18
19     stack->top = -1;
20     stack->capacity = capacity;
21     stack->array = (int*) malloc(stack->capacity * sizeof(int));
22
23     if (!stack->array) return NULL;
24
25     return stack;
26 }
27
28 int isEmpty(struct Stack* stack)
29 {
30     return stack->top == -1 ;
31 }
32
33 int peek(struct Stack* stack)
34 {
35     return stack->array[stack->top];
36 }
37
38 int pop(struct Stack* stack)
39 {
40     if (!isEmpty(stack))
41         return stack->array[stack->top--] ;
42     return '$';
43 }

```

The left sidebar shows the project structure with 'LA2' as the workspace, containing 'Sources' and 'main.c'.

```

44
45 void push(struct Stack* stack,int op)
46 {
47     stack->array[++stack->top] = op;
48 }
49
50 int evaluatePostfix(char* exp)
51 {
52     struct Stack* stack = createStack(strlen(exp));
53     int i;
54
55     if (!stack) return -1;
56
57     for (i = 0; exp[i]; ++i)
58     {
59         if(exp[i]==' ')continue;
60
61         else if (isdigit(exp[i]))
62         {
63             int num=0;
64
65             while(isdigit(exp[i]))
66             {
67                 num=num*10 + (int) (exp[i]-'0');
68                 i++;
69             }
70             i--;
71
72             push(stack,num);
73         }
74
75         else
76         {
77             int val1 = pop(stack);
78             int val2 = pop(stack);
79
80             switch (exp[i])
81             {
82                 case '+': push(stack, val2 + val1); break;
83                 case '-': push(stack, val2 - val1); break;
84                 case '*': push(stack, val2 * val1); break;
85                 case '/': push(stack, val2/val1); break;
86
87             }
88         }
89     }
90     return pop(stack);
91 }
92
93 int main()
94 {
95     char exp[] = "(- (1*2)/3)^14+110-220*16.78*346.621";
96     printf ("%d", evaluatePostfix(exp));
97     return 0;
98 }
99

```

Logs & others

Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++

```

Set Variable: PATH=.;C:\Program Files\CodeBlocks\mingw\bin;C:\Program Files\CodeBlocks\mingw\bin\Program
Files (x86)\Common Files\Intel\Shared Libraries\redist\intel64\compiler;C:\Windows\System32;C:\Windows;C:
\Windows\System32\wbem;C:\Windows\System32\WindowsPowerShell\v1.0;C:\Windows\System32\OpenSSH;C:\Program
Files (x86)\NVIDIA Corporation\PhysX\Common;C:\OrCAD\OrCAD_16.6_Lite\tools\pspice;C:\OrCAD\OrCAD_16.6_Lite
\tools\capture;C:\OrCAD\OrCAD_16.6_Lite\tools\bin;C:\OrCAD\OrCAD_16.6_Lite\OpenAccess\bin\Win32\opt;C:
\OrCAD\OrCAD_16.6_Lite\tools\Fet\bin;C:\OrCAD\OrCAD_16.6_Lite\tools\pcb\bin;C:\Program Files\MikTeX 2.9
\miktex\bin\x64;C:\Program Files\Git\cmd;C:\Program Files\nodejs;C:\Program Files\MATLAB\R2020a\runtime

```

C/C++

Windows (CR+LF)

WINDOWS-1252

Line 19, Col 21, Pos 340

Insert

Modified

Read/Write

default

US



"C:\Users\Sharadindu\Desktop\DSA Lab Assignments\LA2\bin\Debug\LA2.exe"

621

Process returned 0 (0x0) execution time : 0.083 s

Press any key to continue.