Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085

SSN College of Engineering

Department of Computer Science and Engineering

UCS2312 – Data Structures Lab

II Year CSE - B Section (III Semester)

Academic Year 2022-23

Staff Incharge: Dr.H. Shahul Hamead

Exercise-5: Exercises on Stacks

Aim:

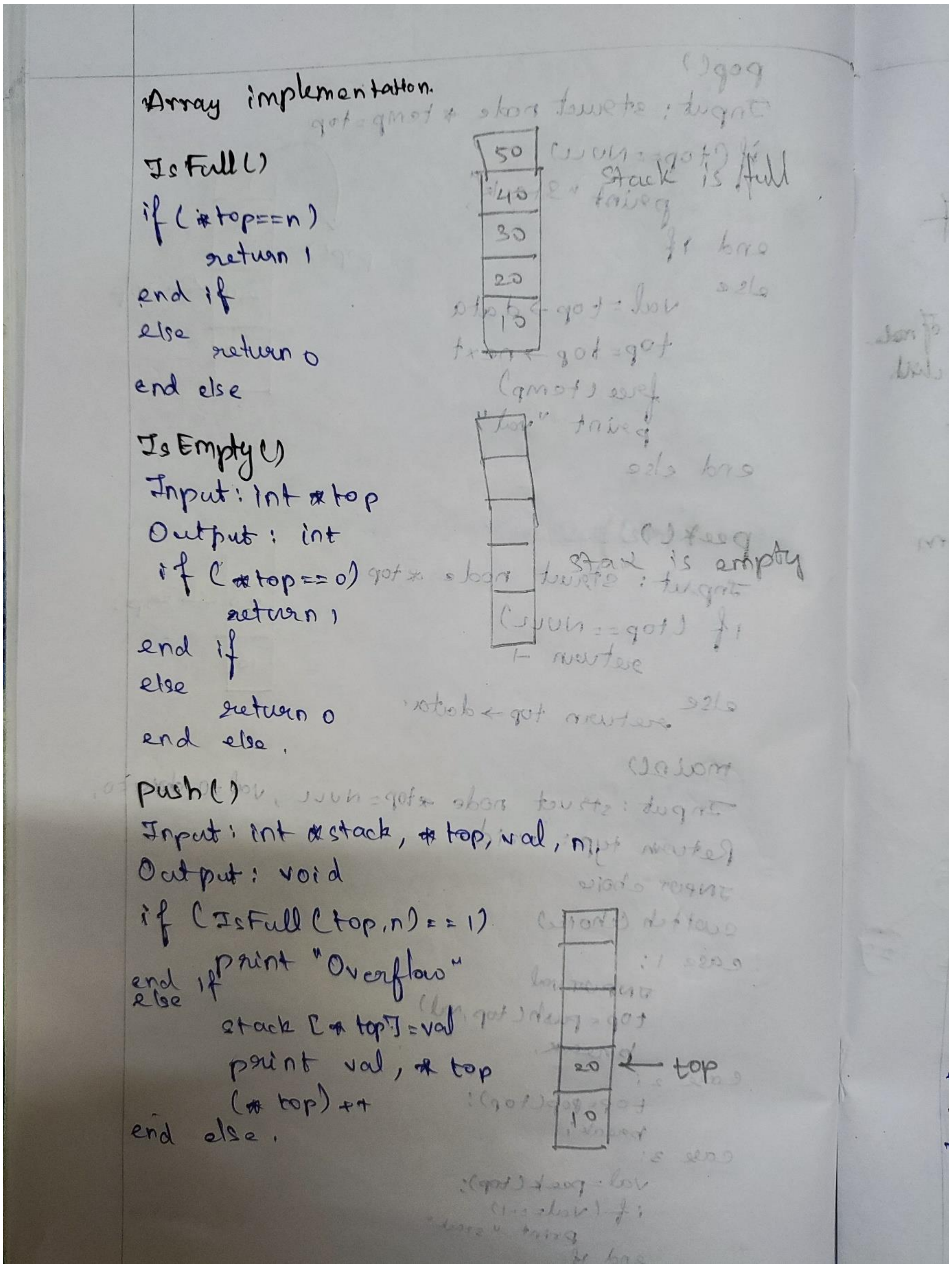
To implement C program in Data structures using the concept of Stacks using linked list and arrays.

Basic

Implement Array version and Linked List version of stacks. Relevant Linked List files can be reused for this problem.

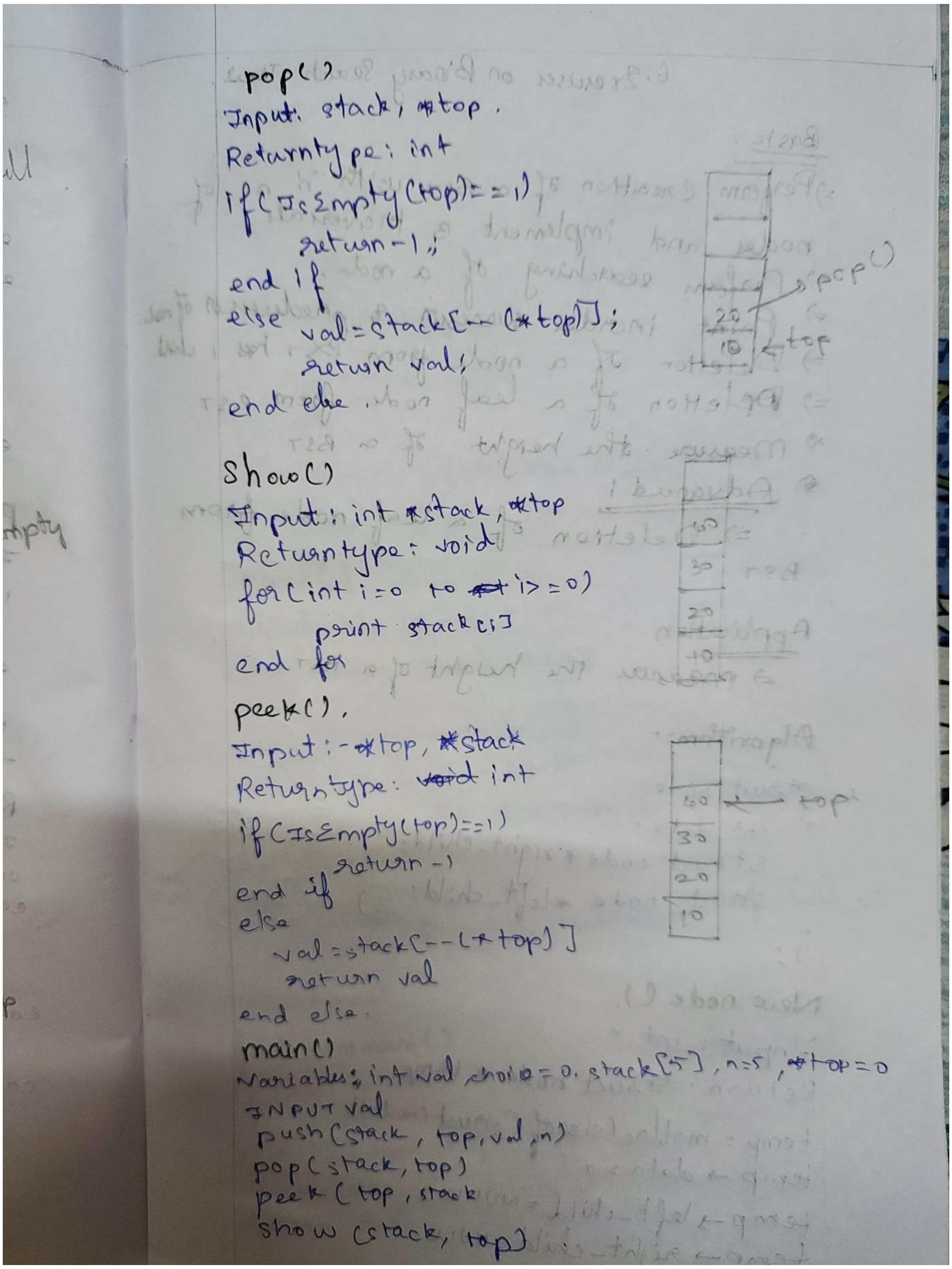
Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085

Pseudocode:

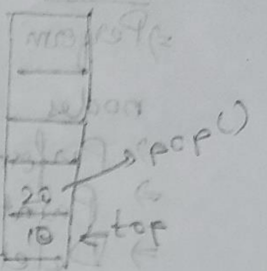

50
40
30
20
10
Stack is full
Stack is empty
20
10
top

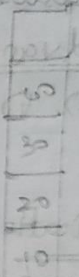
Array implementation.

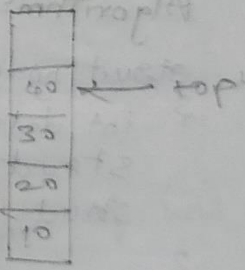
```
IsFull()
if (*top==n)
        return 1
end if
else
        return 0
end else

IsEmpty()
 Input: int *top
 Output: int
  if (*top==0)
        return 1
end if
else
        return 0
end else.

push()
Input: int *stack, *top, val, n
Output: void
if (IsFull(top,n)==1)
        print "Overflow"
end if
else
        stack[*top]=val
        print val, *top
        (*top)++
end else.
```



```
pop()
Input: stack, ~~*~~*top.
Returntype: int
if(IsEmpty(top)==1)
        return -1;
end if
else    val=stack[--(*top)];
        return val;
end else.

Show()
Input: int *stack, *top
Returntype: void
for(int i=0 to ~~*t~~ i>=0)
        print stack[i]
end for

peek().
Input:- *top, *stack
Returntype: ~~void~~ int
if(IsEmpty(top)==1)
        return -1
end if
else
    val=stack[--(*top)]
    return val
end else.
main()
Variables; int val, choice=0, stack[5], n=5, *top=0
INPUT val
 push(stack, top, val, n)
 pop(stack, top)
 peek(top, stack
 show(stack, top)
```







Ex no. 4 Name: M.Rohith

Date: 16-11-22 3122 21 5001 085

Program code:

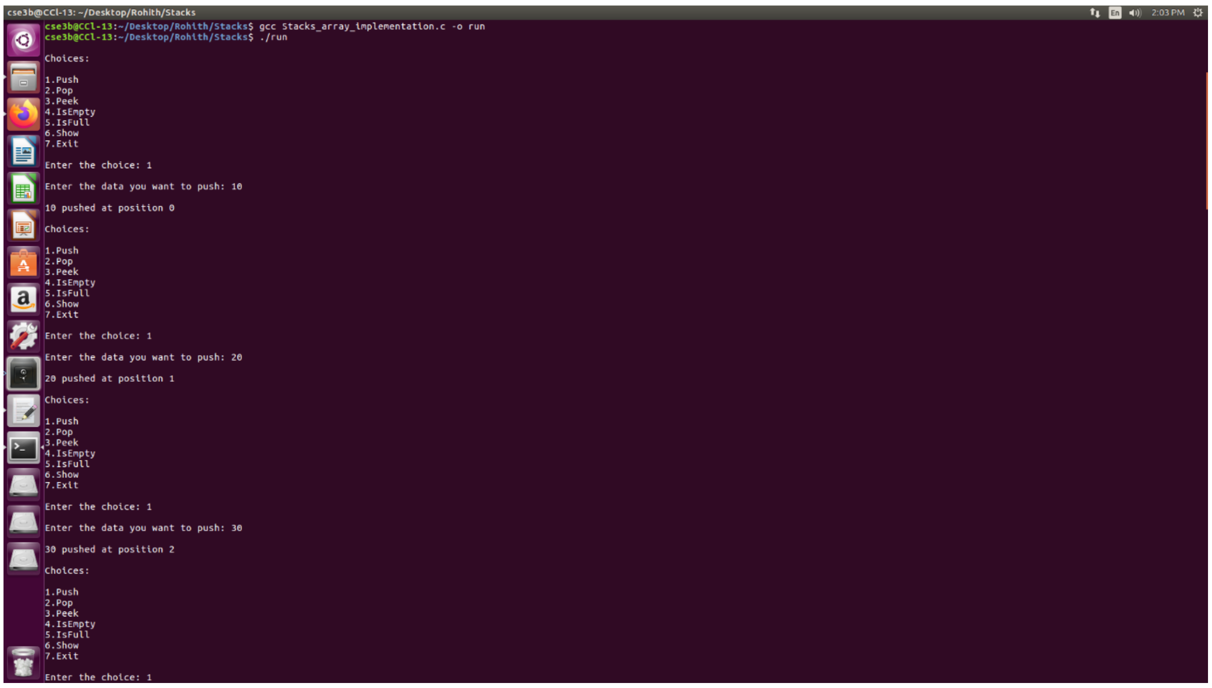
```
#include <stdio.h>
#include <stdlib.h>

int IsFull(int *top,int n)
{
    if(*top==n)
        return 1;
    else
        return 0;
}

int IsEmpty(int *top)
{
    if(*top==0)
        return 1;
    else
        return 0;
}

void push(int *stack,int *top,int val,int n)
{
    if(IsFull(top,n)==1)
        printf("\nStack overflow...\n");
    else
    {
        stack[*top]=val;
        printf("\n%d pushed at position %d\n",val,*top);
        (*top)++;
    }
}

int pop(int *stack,int *top)
{
    int val;
    if(IsEmpty(top)==1)
        return -1;
    else
    {
        val=stack[--(*top)];
        return val;
```

```c
    }
}

void show(int *stack,int *top)
{
    printf("\nDisplaying values in the stack...\n");
    for(int i=(*top)-1;i>=0;i--)
        printf("\n%d found at index %d",stack[i],i);
    if(*top==0)
        printf("\nStack is Empty\n");
    printf("\n");
}

int peek(int *top,int *stack)
{
    int val;
    if(IsEmpty(top)==1)
        return -1;
    else
    {
        val=stack[--(*top)];
        return val;
    }
}

void main()
{
    int val=0,choice=0,stack[5],n=5;
    int *top=(int *)malloc(sizeof(int));
    *top=0;
    do
    {
        printf("\nChoices:\n");
        printf("\n1.Push\n2.Pop\n3.Peek\n4.IsEmpty\n5.IsFull\n6.Show
\n7.Exit\n");
        printf("\nEnter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("\nEnter the data you want to push: ");
                scanf("%d",&val);
                push(stack,top,val,n);
                break;
            case 2:
                val=pop(stack,top);
                if(val==-1)
                    printf("\nStack Underflow...\n");
                else
                    printf("\nPopped value is %d\n",val);
                break;
            case 3:
                val=peek(top,stack);
                if(val==-1)
                    printf("\nStack is Empty\n");
                else
                    printf("\nPeek value is: %d\n",val);
                break;
            case 4:
                if(IsEmpty(top)==1)
                    printf("\nStack is Empty\n");
                else
                    printf("\nStack is not empty\n");
                break;
            case 5:
                if(IsFull(top,n)==1)
                    printf("\nStack is Full\n");
                else
                    printf("\nStack is not full\n");
                break;
            case 6:
                show(stack,top);
                break;
            case 7:
                exit(0);
            default:
                printf("\nInvalid choice...\n");
        }
    }while(choice!=7);
}
```

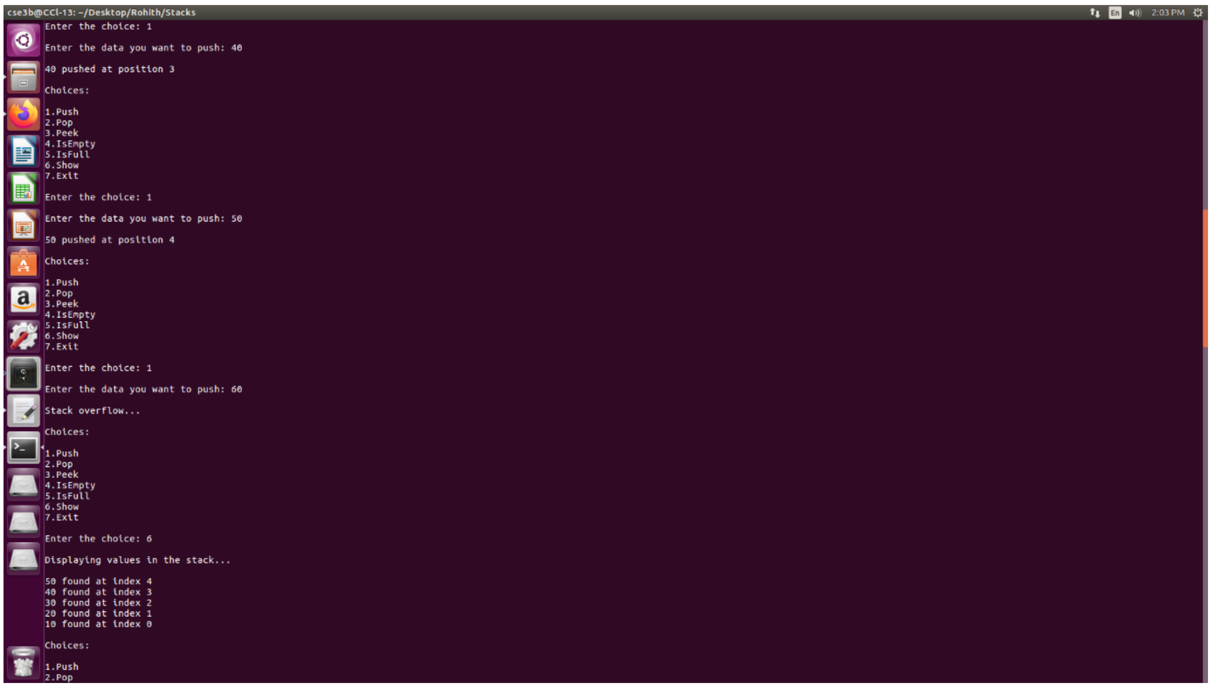
Ex no. 4 Name: M.Rohith

Date: 16-11-22 3122 21 5001 085

Output:

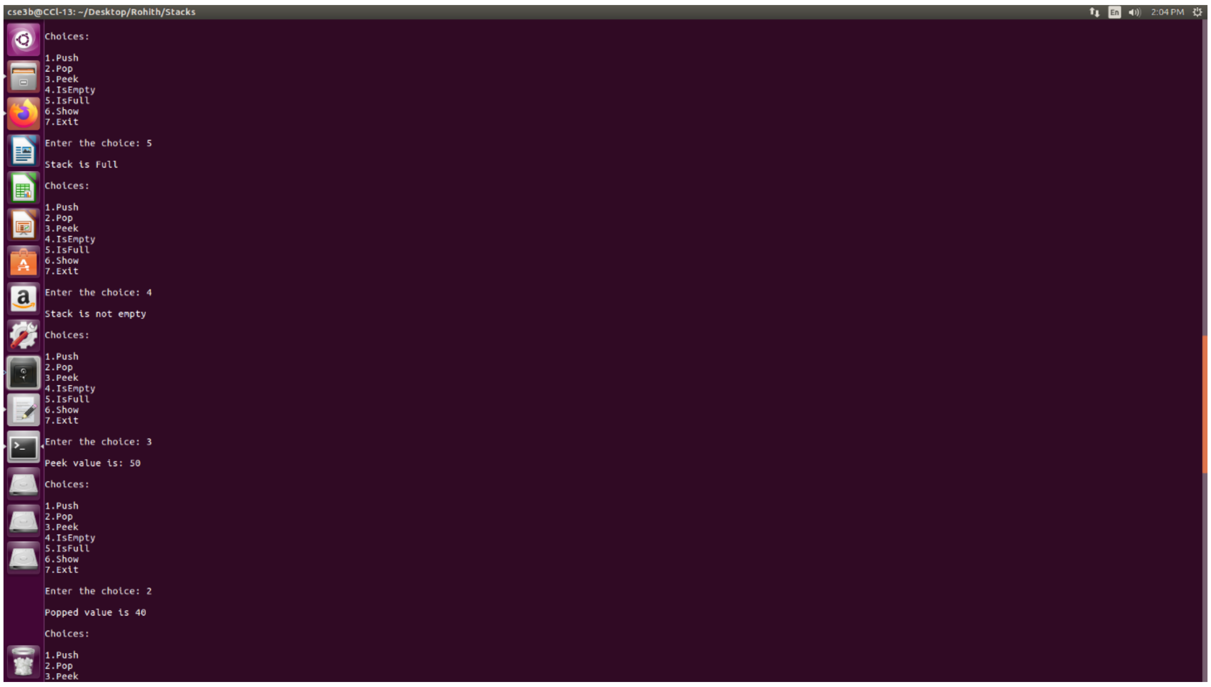


```
cse3b@CCl-13: ~/Desktop/Rohith/Stacks                                   2:03 PM
cse3b@CCl-13:~/Desktop/Rohith/Stacks$ gcc Stacks_array_implementation.c -o run
cse3b@CCl-13:~/Desktop/Rohith/Stacks$ ./run

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 1

Enter the data you want to push: 10

10 pushed at position 0

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 1

Enter the data you want to push: 20

20 pushed at position 1

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 1

Enter the data you want to push: 30

30 pushed at position 2

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 1
```

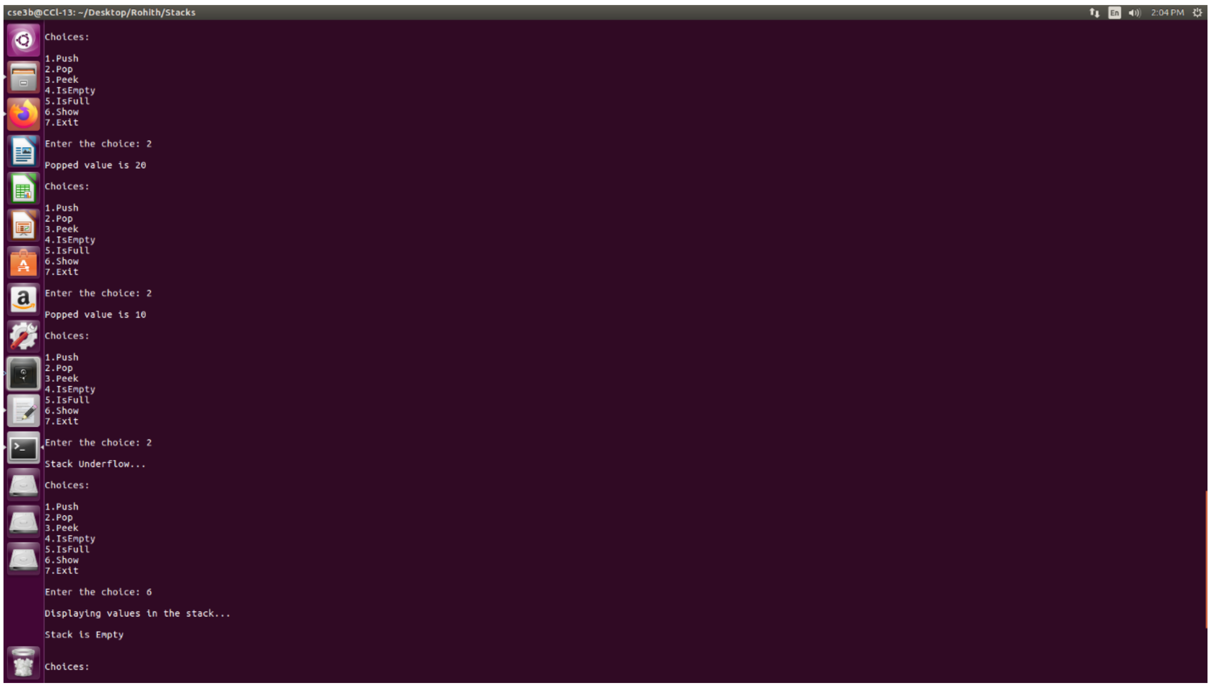


```
cse3b@CCl-13: ~/Desktop/Rohith/Stacks                                   2:03 PM
Enter the choice: 1

Enter the data you want to push: 40

40 pushed at position 3

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 1

Enter the data you want to push: 50

50 pushed at position 4

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 1

Enter the data you want to push: 60

Stack overflow...

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 6

Displaying values in the stack...

50 found at index 4
40 found at index 3
30 found at index 2
20 found at index 1
10 found at index 0

Choices:

1.Push
2.Pop
```

Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085



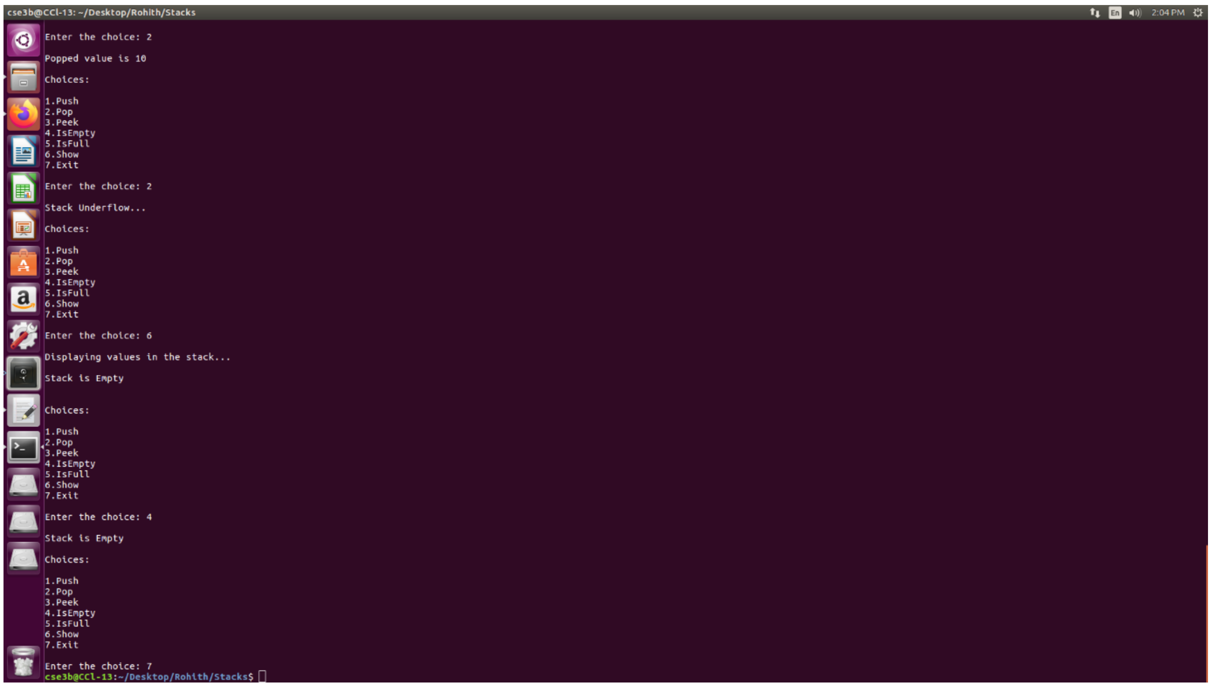
```
cse3b@CCl-13: ~/Desktop/Rohith/Stacks                2:04 PM
Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 5

Stack is Full

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 4

Stack is not empty

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 3

Peek value is: 50

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 2

Popped value is 40

Choices:

1.Push
2.Pop
3.Peek
```



```
cse3b@CCl-13: ~/Desktop/Rohith/Stacks                2:04 PM
Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 2

Popped value is 20

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 2

Popped value is 10

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 2

Stack Underflow...

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 6

Displaying values in the stack...

Stack is Empty

Choices:
```

Ex no. 4
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

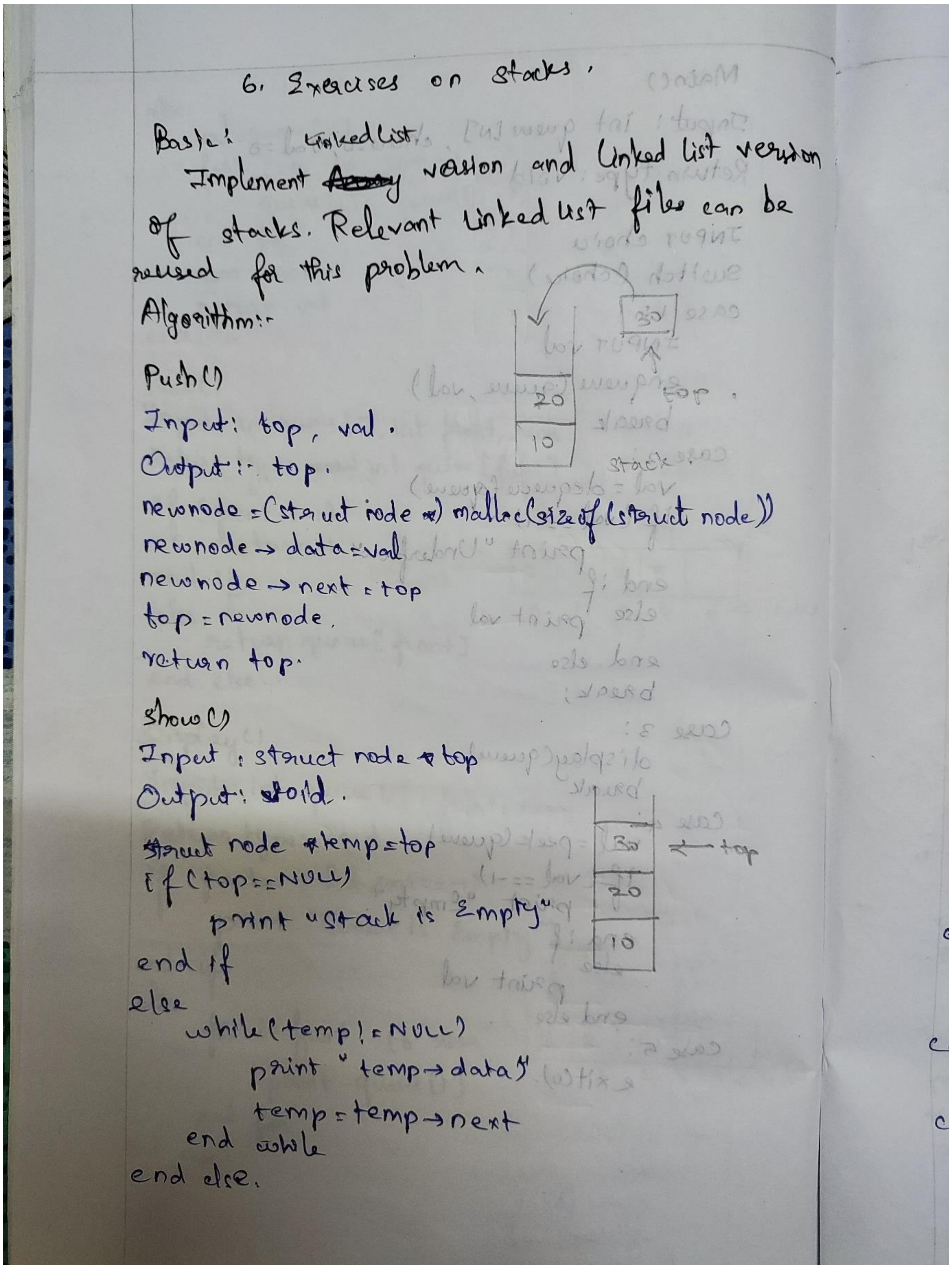


```
cse3b@CCl-13: ~/Desktop/Rohith/Stacks
Enter the choice: 2

Popped value is 10

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 2

Stack Underflow...

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 6

Displaying values in the stack...

Stack is Empty


Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 4

Stack is Empty

Choices:

1.Push
2.Pop
3.Peek
4.IsEmpty
5.IsFull
6.Show
7.Exit

Enter the choice: 7
cse3b@CCl-13:~/Desktop/Rohith/Stacks$
```

Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085

Linked List version:

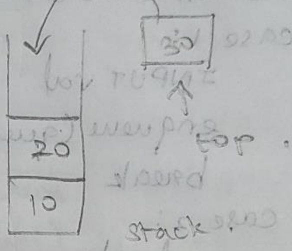
Pseudocode:



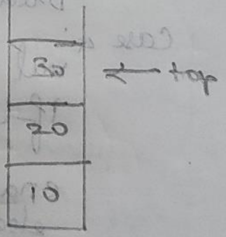
6. Exercises on stacks.

Basic: ~~Linked List~~
Implement ~~Array~~ version and Linked list version of stacks. Relevant Linked list files can be reused for this problem.

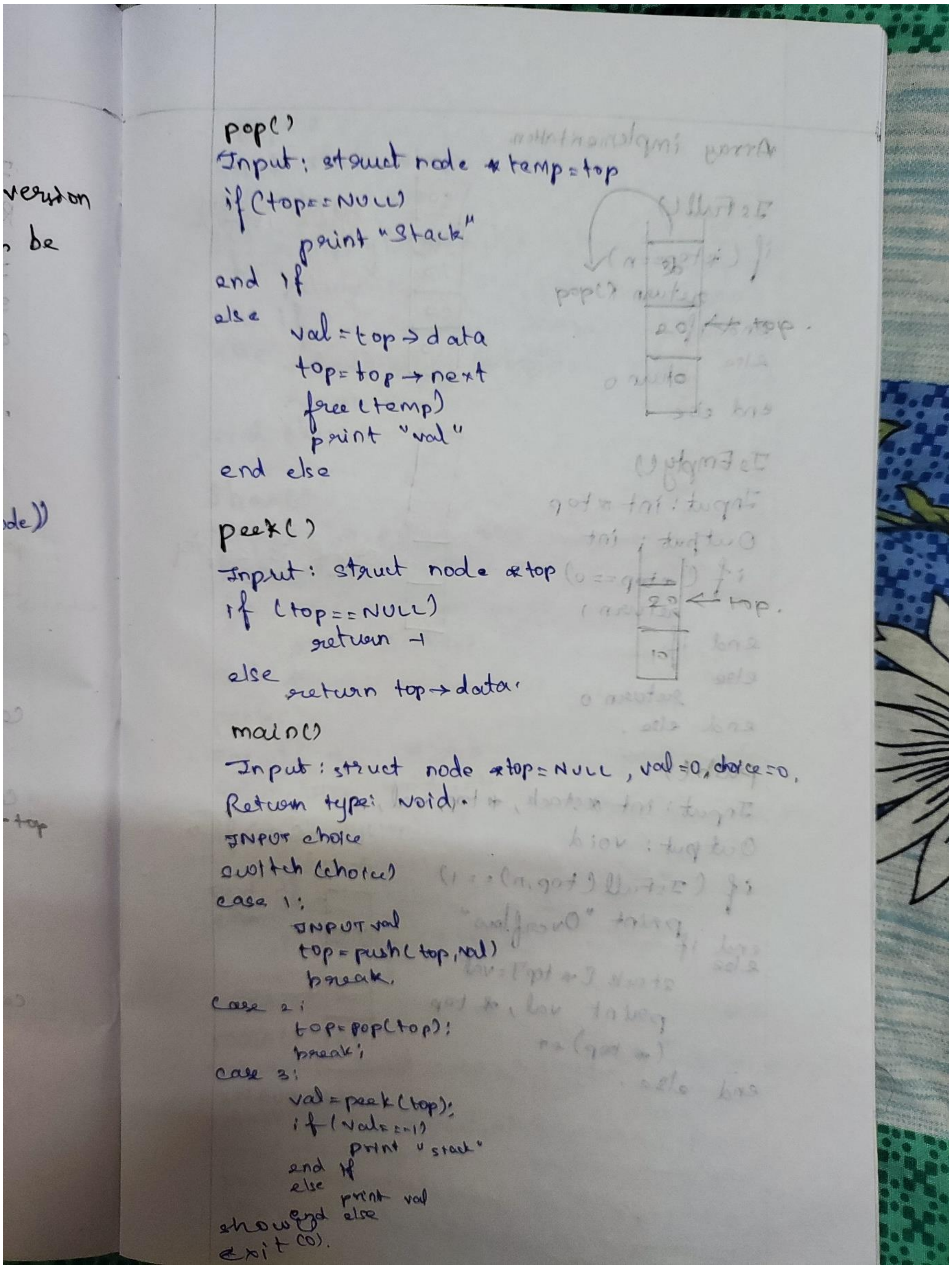
Algorithm:-



```
Push()
Input: top, val.
Output:- top.
newnode =(struct node *) malloc(sizeof(struct node))
newnode -> data=val
newnode -> next = top
top = newnode.
return top.

show()
Input : struct node * top
Output: ~~s~~void.
~~struct~~ node *temp=top
if(top==NULL)
    print "stack is Empty"
end if
else
    while(temp!=NULL)
        print "temp->data"
        temp=temp->next
    end while
end else.
```



Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085



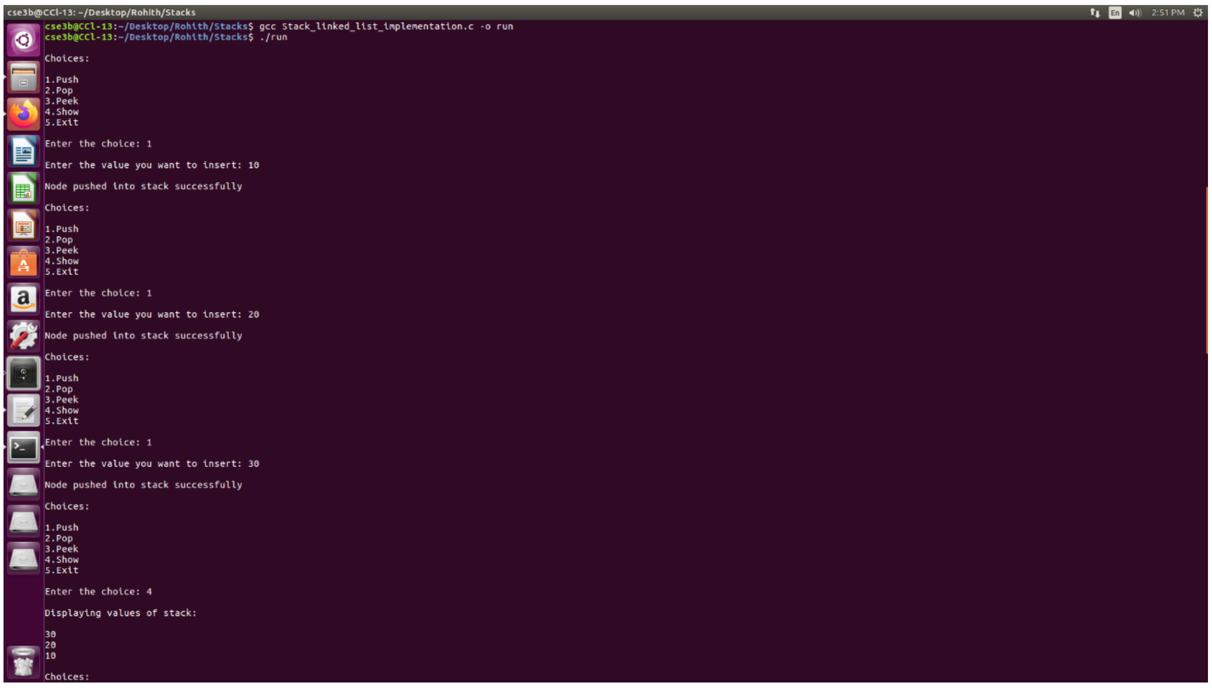
```
pop()
Input: struct node * temp=top
if(top==NULL)
        print "Stack"
end if
else
        val=top->data
        top=top->next
        free(temp)
        print "val"
end else

peek()
Input: struct node *top
if (top==NULL)
        return -1
else
        return top->data.

main()
Input: struct node *top=NULL, val=0, choice=0.
Return type: void.
INPUT choice
switch (choice)
case 1:
        INPUT val
        top=push(top,val)
        break.
Case 2:
        top=pop(top);
        break;
case 3:
        val=peek(top);
        if(val==-1)
            print "stack"
        end if
        else
            print val
        end else
show
exit (0).
```

Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085

Program code:

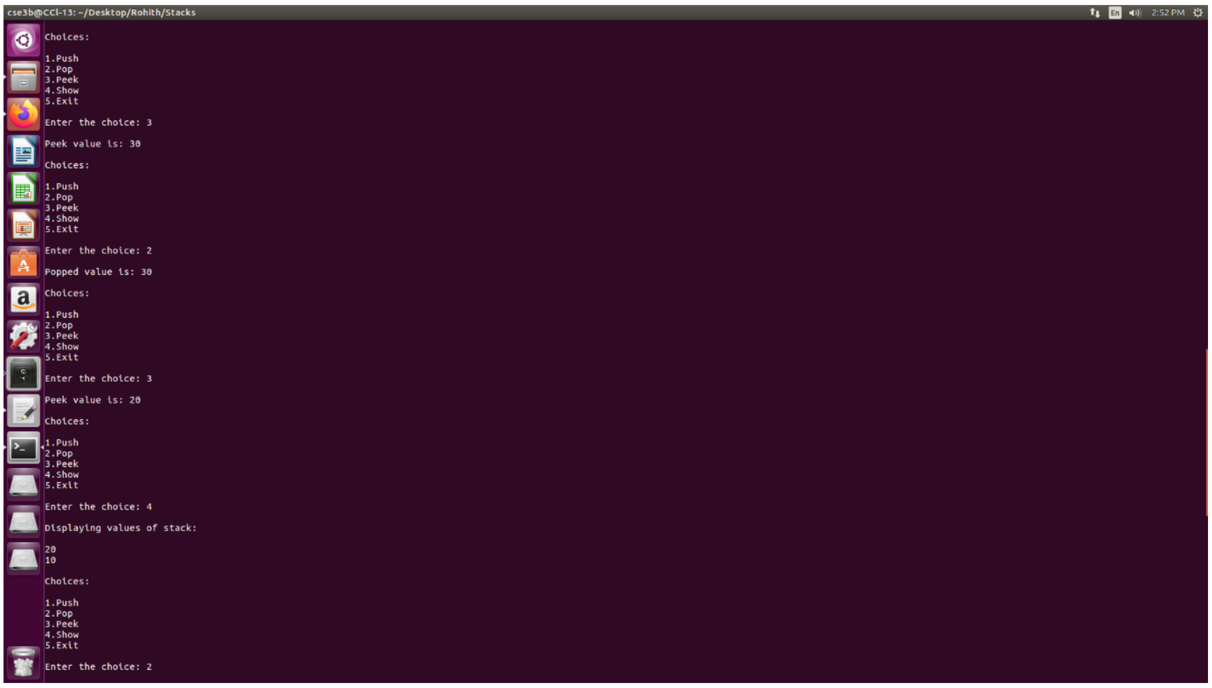
```c
#include <stdio.h>
#include <stdlib.h>

struct node
{
    int data;
    struct node *next;
};

struct node* push(struct node *top,int val)
{
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=val;
    newnode->next=top;
    top=newnode;
    printf("\nNode pushed into stack successfully\n");
    return top;
}

void show(struct node *top)
{
    struct node *temp=top;
    printf("\nDisplaying values of stack:\n");
    if(top==NULL)
        printf("\nStack is Empty\n");
    else
    {
        while(temp!=NULL)
        {
            printf("\n%d",temp->data);
            temp=temp->next;
        }
        printf("\n");
    }
}

struct node* pop(struct node *top)
{
    struct node *temp=top;
    int val=0;
    if(top==NULL)
        printf("\nStack Underflow\n");
    else
    {
        val=top->data;
        top=top->next;
        free(temp);
        printf("\nPopped value is: %d\n",val);
    }
    return top;
}

int peek(struct node *top)
{
    if(top==NULL)
        return -1;
    else
        return top->data;
}

void main()
{
    struct node *top=NULL;
    int val=0,choice=0;
    do
    {
        printf("\nChoices:\n");
        printf("\n1.Push\n2.Pop\n3.Peek\n4.Show\n5.Exit\n");
        printf("\nEnter the choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("\nEnter the value you want to insert: ");
                scanf("%d",&val);
                top=push(top,val);
                break;
            case 2:
                top=pop(top);
                break;
            case 3:
                val=peek(top);
                if(val==-1)
                    printf("\nStack is empty\n");
                else
                    printf("\nPeek value is: %d\n",val);
                break;
            case 4:
                show(top);
                break;
            case 5:
                exit(0);
            default:
                printf("\nInvalid choice\n");
        }
    }while(choice!=5);
}
```

Output:

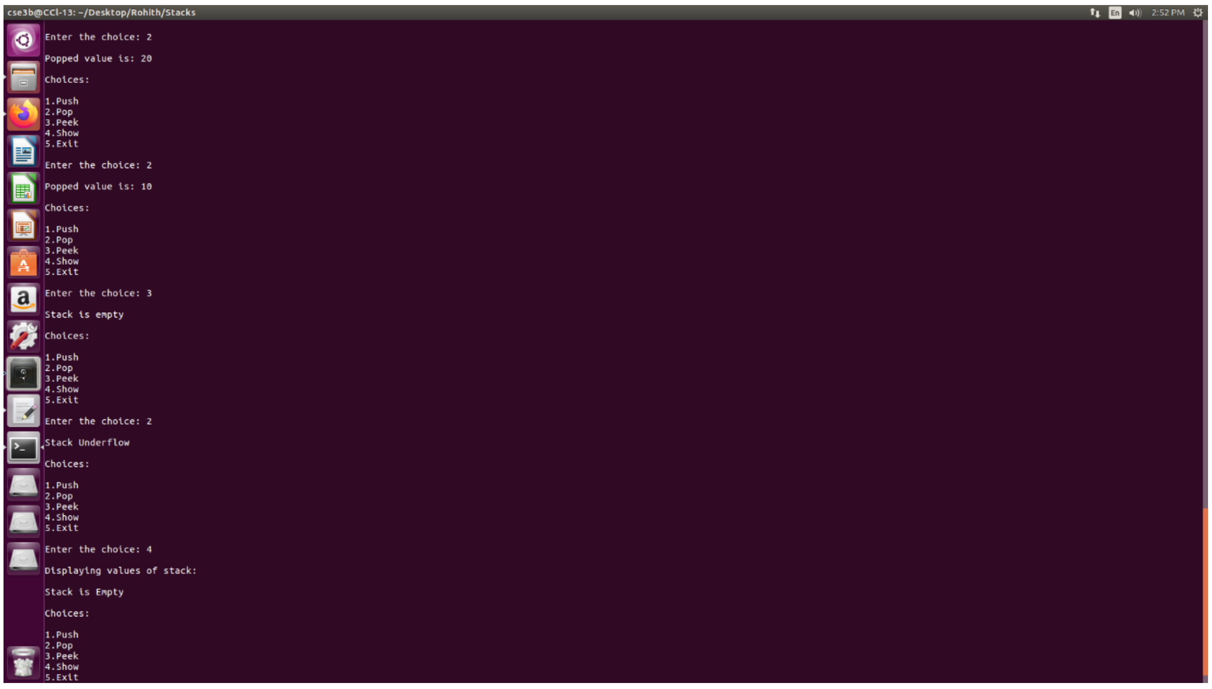


```
cse3b@CCl-13:~/Desktop/Rohith/Stacks$ gcc Stack_linked_list_implementation.c -o run
cse3b@CCl-13:~/Desktop/Rohith/Stacks$ ./run

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 1

Enter the value you want to insert: 10

Node pushed into stack successfully

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 1

Enter the value you want to insert: 20

Node pushed into stack successfully

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 1

Enter the value you want to insert: 30

Node pushed into stack successfully

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 4

Displaying values of stack:

30
20
10

Choices:
```

Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085



```
cse3b@CCl-13: ~/Desktop/Rohith/Stacks
Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 3

Peek value is: 30

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 2

Popped value is: 30

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 3

Peek value is: 20

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 4

Displaying values of stack:

20
10

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 2
```



```
cse3b@CCl-13: ~/Desktop/Rohith/Stacks
Enter the choice: 2

Popped value is: 20

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 2

Popped value is: 10

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 3

Stack is empty

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 2

Stack Underflow

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit

Enter the choice: 4

Displaying values of stack:

Stack is Empty

Choices:

1.Push
2.Pop
3.Peek
4.Show
5.Exit
```

Ex no. 4 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085

Application:

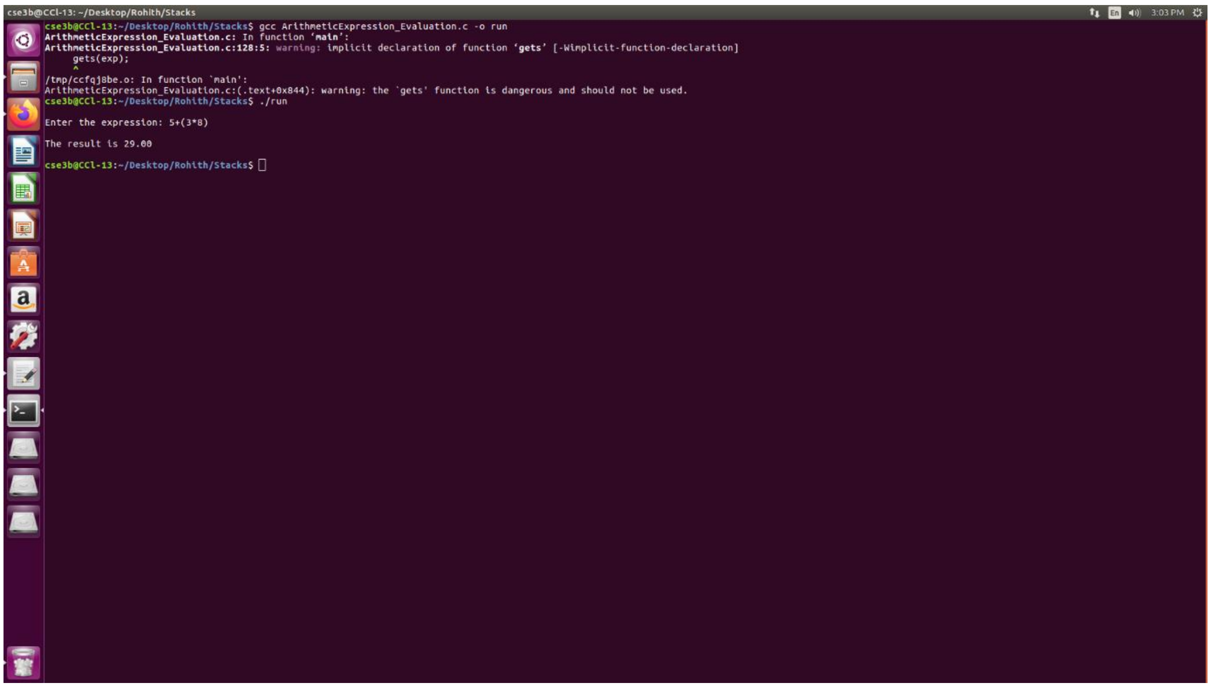
Implement a calculator which takes arithmetic expression as input and performs the operation.

Program code:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
float calculate(char* ptr);
void push(char stack[], int n, int* top, char data){
    if (*top >= n){
        printf("Stack overflow!");
    }
    else{
        stack [++(*top)] = data;
    }
}
char pop(char stack[], int* top){
    if (*top == -1){
        printf("The stack is empty!");
    }
    else{
        char ch = stack[(*top)];
        (*top)--;
        return ch;
    }
}
void push2(float stack[], int n, int* top, float data){
    if (*top >= n){
        printf("Stack overflow!");
    }
    else{
        stack [++(*top)] = data;
    }
}
float pop2(float stack[], int* top){
    if (*top == -1){
        printf("The stack is empty!");
    }
    else{
        float f = stack[(*top)];
        (*top)--;
        return f;
    }
}
int priority(char c){
    if (c == '(')
        return 0;
    if (c == '+' || c == '-')
        return 1;
    if (c == '/' || c == '*')
        return 2;
    return 0;
}
char* convertType(char stack[], int n, int* top, char* a){
    char* arr = (char*)malloc(sizeof(char*));
    char c, ch[100];
    int i=0;
    while (*a != '\0'){
        if (isalnum(*a)){
            ch[i++] = *a;
        }
        else if (*a == '('){
            push(stack, n, top, *a);
        }
        else if (*a == ')'){
            while ((c = pop(stack, top)) != '('){
                ch[i++] = c;
            }
        }
        else {
            while (priority(stack[*top]) >= priority(*a)){
                ch[i++] = pop(stack, top);
            }
            push(stack, n, top, *a);
        }
        a++;
    }
    while (*top != -1){
        ch[i++] = pop(stack, top);
```

```c
    }
    arr = ch;
    printf("\nThe result is %.2f \n\n", calculate(arr));
    return arr;
}
float calculate(char* ptr){
    float result;
    char c = *ptr;
    float stack2[100];
    int *t = (int*)malloc(sizeof(int*));
    *t = -1;
    int n = 20;
    while (c != '\0'){
        if ( c >= '0' && c <= '9'){
            push2(stack2, n, t, c - '0');
        }
        else if (c == '+'){
            float a = pop2(stack2, t);
            float b = pop2(stack2, t);
            float sum = a+b;
            push2(stack2, n, t, sum);
        }
        else if (c == '-'){
            float a = pop2(stack2, t);
            float b = pop2(stack2, t);
            float diff = b-a;
            push2(stack2, n, t, diff);
        }
        else if (c == '*'){
            float a = pop2(stack2, t);
            float b = pop2(stack2, t);
            float mul = a*b;
            push2(stack2, n, t, mul);
        }
        else if (c == '/'){
            float a = pop2(stack2, t);
            float b = pop2(stack2, t);
            float div = b/a;
            push2(stack2, n, t, div);
        }
        c = *(ptr++);
    }
    result = pop2(stack2, t);
    return result;
}
void main(){
    int *top = (int*)malloc(sizeof(int*));
    char *arr =  (char*)malloc(sizeof(char*));
    *top = -1;
    int n=20, i=0;
    char stack[10], *a, exp[100];
    printf("\nEnter the expression: ");
    gets(exp);
    a = exp;
    arr = convertType(stack, n, top, a);
}
```

Output:



```
cse3b@CCl-13:~/Desktop/Rohith/Stacks
cse3b@CCl-13:~/Desktop/Rohith/Stacks$ gcc ArithmeticExpression_Evaluation.c -o run
ArithmeticExpression_Evaluation.c: In function ‘main’:
ArithmeticExpression_Evaluation.c:128:5: warning: implicit declaration of function ‘gets’ [-Wimplicit-function-declaration]
     gets(exp);
     ^
/tmp/ccfqj8be.o: In function `main':
ArithmeticExpression_Evaluation.c:(.text+0x844): warning: the `gets' function is dangerous and should not be used.
cse3b@CCl-13:~/Desktop/Rohith/Stacks$ ./run

Enter the expression: 5+(3*8)

The result is 29.00

cse3b@CCl-13:~/Desktop/Rohith/Stacks$ 
```

Result:

Hence C program using stacks data structures has been implemented in both array and linked list versions.