WEEK-3 MODULE

NAME:MD.Amrath **ID**:B181094 **CLASS**:AB2-305

1.Linear Search:

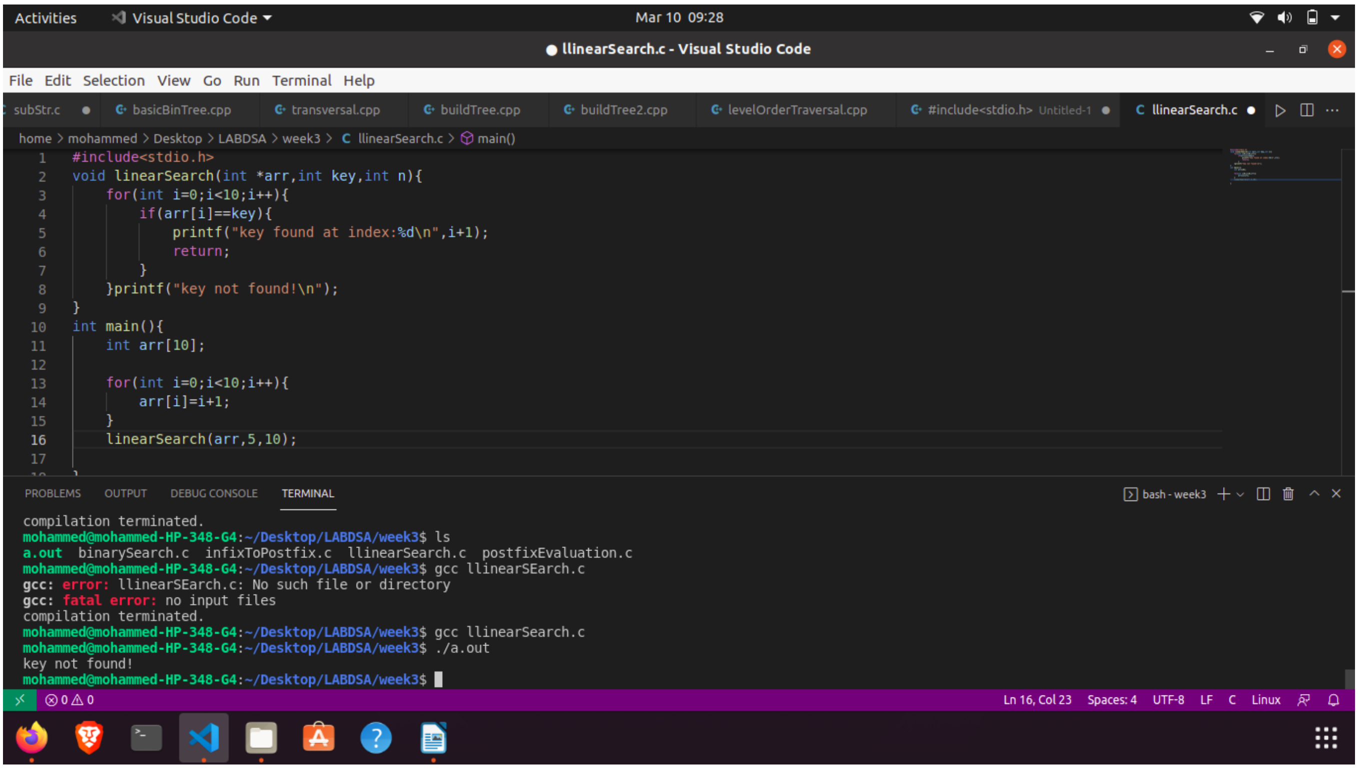
PSEUDOCODE:

```
procedure linear_search (list, value)

   for each item in the list
      if match item == value
         return the item's location
      end if
   end for

end procedure
```

CODE:

```
#include<stdio.h>
void linearSearch(int *arr,int key,int n){
    for(int i=0;i<10;i++){
        if(arr[i]==key){
            printf("key found at index:%d\n",i+1);
            return;
        }
    }printf("key not found!\n");
}
int main(){
    int arr[10];

    for(int i=0;i<10;i++){
        arr[i]=i+1;
    }
    linearSearch(arr,20,10);

}
```

OUTPUT:



```c
#include<stdio.h>
void linearSearch(int *arr,int key,int n){
    for(int i=0;i<10;i++){
        if(arr[i]==key){
            printf("key found at index:%d\n",i+1);
            return;
        }
    }printf("key not found!\n");
}
int main(){
    int arr[10];

    for(int i=0;i<10;i++){
        arr[i]=i+1;
    }
    linearSearch(arr,5,10);

}
```

```
compilation terminated.
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ ls
a.out  binarySearch.c  infixToPostfix.c  llinearSearch.c  postfixEvaluation.c
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ gcc llinearSEarch.c
gcc: error: llinearSEarch.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ gcc llinearSearch.c
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ ./a.out
key not found!
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ 
```

2.BINARY SEARCH

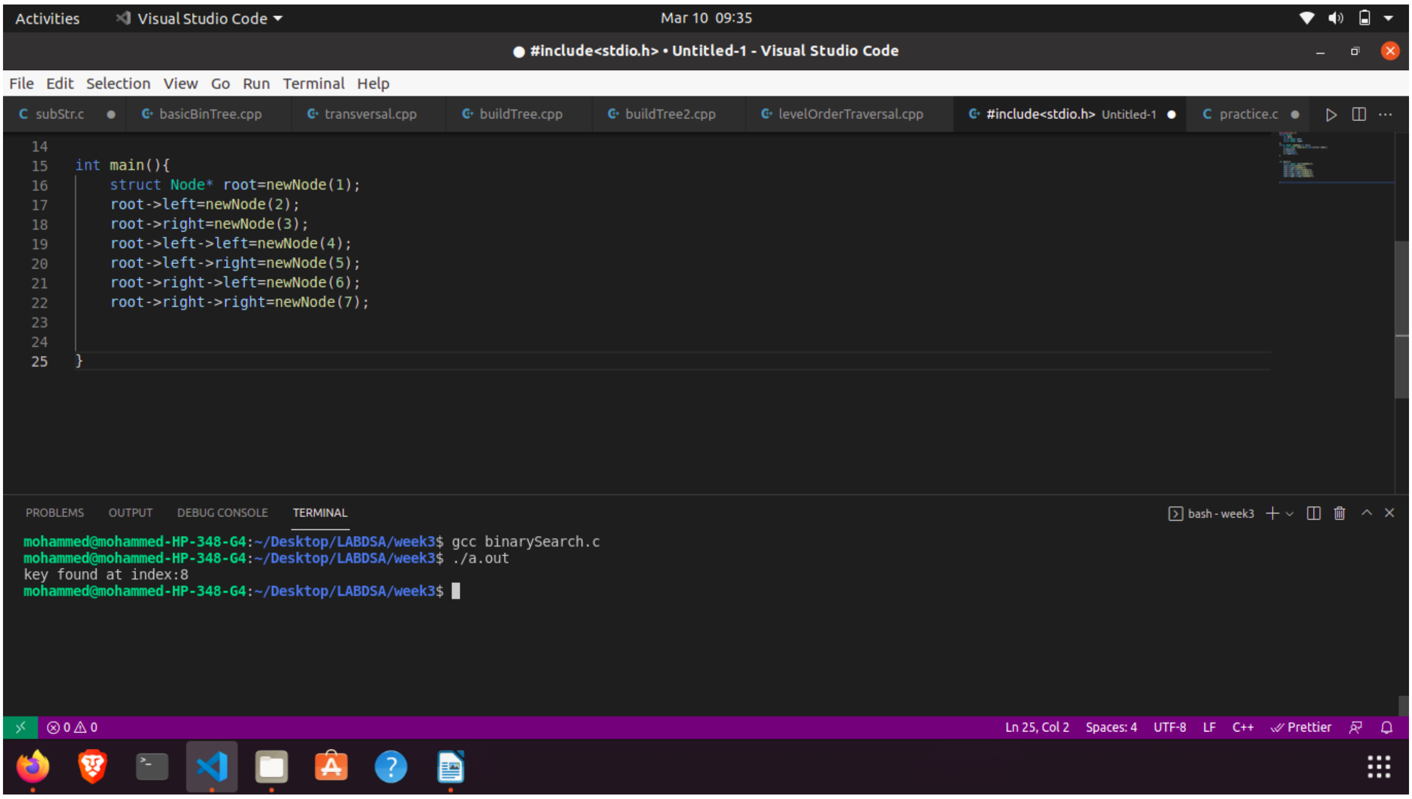
PSEUDOCODE:

```
Procedure binary_search
   A ← sorted array
   n ← size of array
   x ← value to be searched

   Set lowerBound = 1
   Set upperBound = n

   while x not found
      if upperBound < lowerBound
         EXIT: x does not exists.

      set midPoint = lowerBound + ( upperBound - lowerBound ) / 2

      if A[midPoint] < x
         set lowerBound = midPoint + 1

      if A[midPoint] > x
         set upperBound = midPoint - 1

      if A[midPoint] = x
         EXIT: x found at location midPoint
   end while

end procedure
```

CODE:

```
#include<stdio.h>
void binarySearch(int *a,int n,int key){
    int low=0;
    int high=n-1;
    int mid=-1;
    while(low<=high){
        mid=(low+high)/2;
        if(a[mid]<key){
            low=mid+1;
        }
        else if(a[mid]>key){
            high=mid-1;
        }
        else if(a[mid]==key){
            printf("key found at index:%d\n",mid);
            return;
        }
    }printf("key not found\n");
}
int main(){
    int arr[10];
    for(int i=0;i<10;i++){
        arr[i]=i+1;
    }
    binarySearch(arr,10,9);
}
```

OUTPUT:



```
int main(){
    struct Node* root=newNode(1);
    root->left=newNode(2);
    root->right=newNode(3);
    root->left->left=newNode(4);
    root->left->right=newNode(5);
    root->right->left=newNode(6);
    root->right->right=newNode(7);


}
```

```
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ gcc binarySearch.c
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ ./a.out
key found at index:8
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ 
```

3.POSTFIX EVALUATION:

PSEUDOCODE:

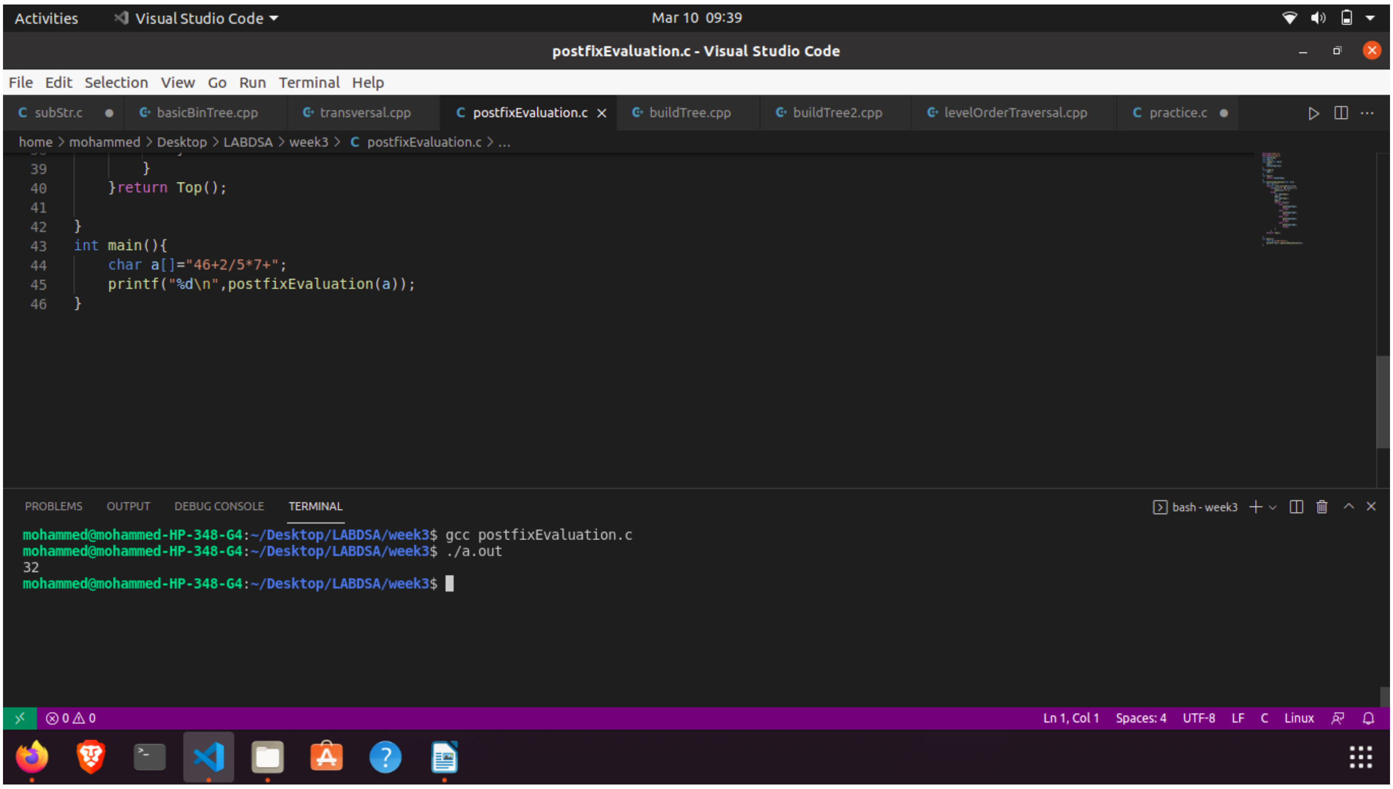
```
Let the given expression be “2 3 1 * + 9 -“. We scan all elements one by one.
1) Scan ‘2’, it’s a number, so push it to stack. Stack contains ‘2’
2) Scan ‘3’, again a number, push it to stack, stack now contains ‘2 3’ (from bottom to top)
3) Scan ‘1’, again a number, push it to stack, stack now contains ‘2 3 1’
4) Scan ‘*’, it’s an operator, pop two operands from stack, apply the * operator on operands, we get 3*1 which results in 3. We push the result ‘3’ to stack. The stack now becomes ‘2 3’.
5) Scan ‘+’, it’s an operator, pop two operands from stack, apply the + operator on operands, we get 3 + 2 which results in 5. We push the result ‘5’ to stack. The stack now becomes ‘5’.
6) Scan ‘9’, it’s a number, we push it to the stack. The stack now becomes ‘5 9’.
7) Scan ‘-‘, it’s an operator, pop two operands from stack, apply the – operator on operands, we get 5 – 9 which results in -4. We push the result ‘-4’ to the stack. The stack now becomes ‘-4’.
8) There are no more elements to scan, we return the top element from the stack (which is the only element left in a stack).
```

CODE:

```
#include<stdio.h>
#include<string.h>
int stack[10];
int top=-1;
void push(int val){
   top++;
   stack[top]=val;
}
void pop(){
   top--;
}
int Top(){
   return stack[top];
}
int postfixEvaluation(char *s){
   char d[]="";
   for(int i=0;i<strlen(s);i++){
      if(s[i]>='0' && s[i]<='9')
         push(s[i]-'0');
      else{
         int op2=Top();
         pop();
         int op1=Top();
         pop();
         switch (s[i]){
            case '+':
               push(op1+op2);
               break;
            case '-':
               push(op1-op2);
               break;
```

```
                case '*':
                    push(op1*op2);
                    break;
                case '/':
                    push(op1/op2);
                    break;
            }
        }
    }return Top();

}
int main(){
    char a[]="46+2/5*7+";
    printf("%d\n",postfixEvaluation(a));
}
```

OUTPUT:


Activities
Visual Studio Code
Mar 10 09:39
postfixEvaluation.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
subStr.c
basicBinTree.cpp
transversal.cpp
postfixEvaluation.c
buildTree.cpp
buildTree2.cpp
levelOrderTraversal.cpp
practice.c
home > mohammed > Desktop > LABDSA > week3 > postfixEvaluation.c > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
bash - week3
⊗ 0 ⚠ 0
Ln 1, Col 1 Spaces: 4 UTF-8 LF C Linux

```
        }
    }return Top();

}
int main(){
    char a[]="46+2/5*7+";
    printf("%d\n",postfixEvaluation(a));
}
```

```
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ gcc postfixEvaluation.c
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ ./a.out
32
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ 
```

3.POSTFIX TO INFIX EVALUATION:

PSEUDOCODE:

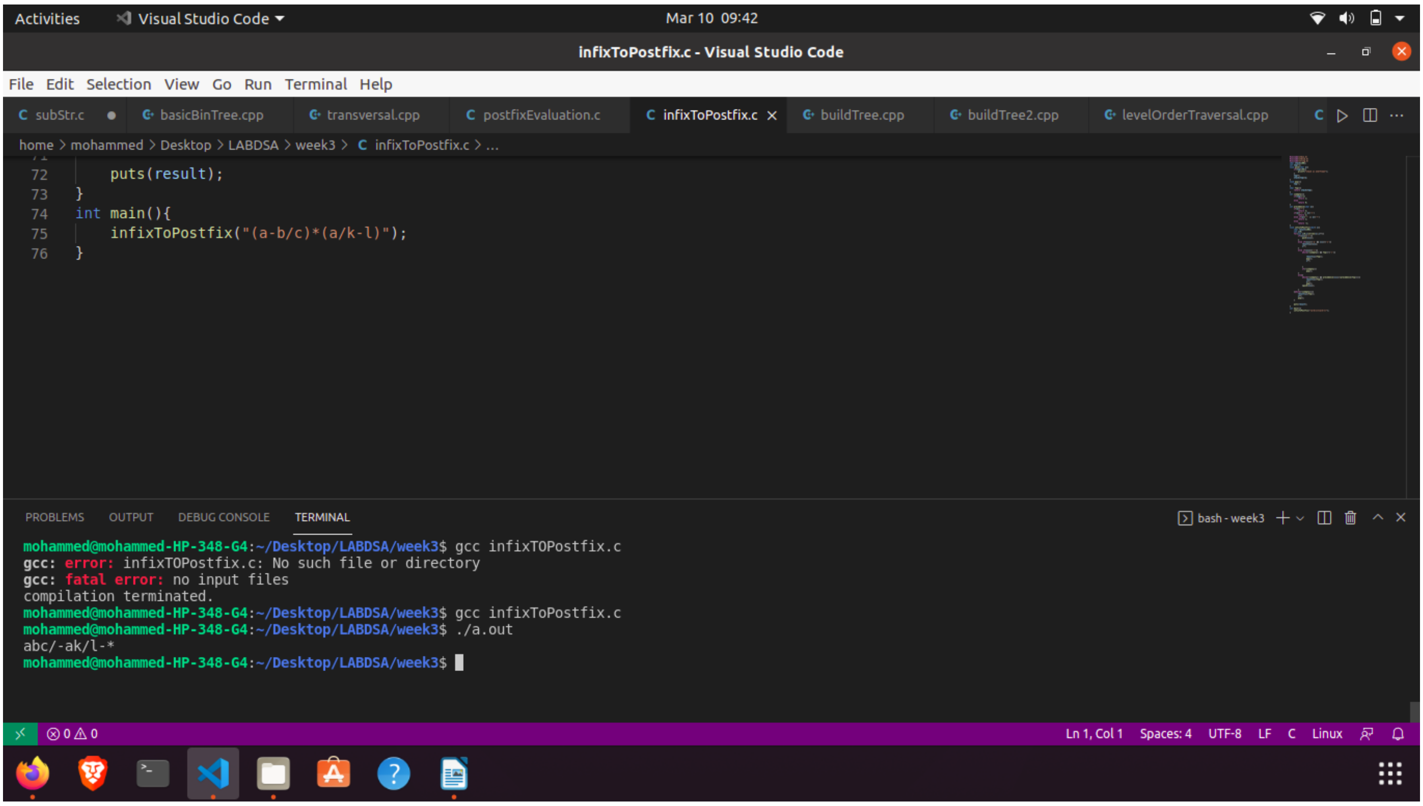
```
Scan input string from left to right character by character.
    1. If the character is an operand, put it into output stack.
    2. If the character is an operator and operator's stack is empty, push operator into operators'
       stack.
```

3. If the operator's stack is not empty, there may be following possibilities.
4. If the precedence of scanned operator is greater than the top most operator of operator's stack, push this operator into operand's stack.
5. If the precedence of scanned operator is less than or equal to the top most operator of operator's stack, pop the operators from operand's stack until we find a low precedence operator than the scanned character. Never pop out ('(') or (')') whatever may be the precedence level of scanned character.
6. If the character is opening round bracket ('('), push it into operator's stack.
7. If the character is closing round bracket (')'), pop out operators from operator's stack until we find an opening bracket ('(').
8. Now pop out all the remaining operators from the operator's stack and push into output stack.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char stack[100];
int top=-1;
void push(char a){
    if(top>=99){
        printf("stack is overflown");
    }
    top++;
    stack[top]=a;
}
void pop(){
    top--;
}
char Top(){
    return stack[top];
}
int isempty(){
    if(top==-1)
        return 1;
    else
        return 0;
}
int precedence(char a){
    if(a=='^')
        return 3;
    if(a=='/' || a=='*')
        return 2;
    else if(a=='-' || a=='+')
        return 1;
    else
        return -1;
}
void infixToPostfix(char* s){
    char result[100];
    int j=0;
    for(int i=0;i<strlen(s);i++){
        if(s[i]=='('){
            push(s[i]);
        }
        else if(s[i]>='a' && s[i]<='z'){
            result[j]=s[i];
            j++;
        }
        else if(s[i]==')'){
            while(!isempty() && Top()!='('){

                result[j]=Top();
                pop();
                j++;


            }
            if(!isempty())
                pop();
        }
        else{
            while(!isempty() && precedence(s[i])<precedence(Top())){
                result[j]=Top();
                j++;
                pop();
            }push(s[i]);

        }
    }while(!isempty()){
        result[j]=Top();
        j++;
        pop();
    }

    puts(result);
}
int main(){
    infixToPostfix("(a-b/c)*(a/k-l)");
}
```

OUTPUT:



```c
    puts(result);
}
int main(){
    infixToPostfix("(a-b/c)*(a/k-l)");
}
```

```
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ gcc infixTOPostfix.c
gcc: error: infixTOPostfix.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ gcc infixToPostfix.c
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ ./a.out
abc/-ak/l-*
mohammed@mohammed-HP-348-G4:~/Desktop/LABDSA/week3$ 
```