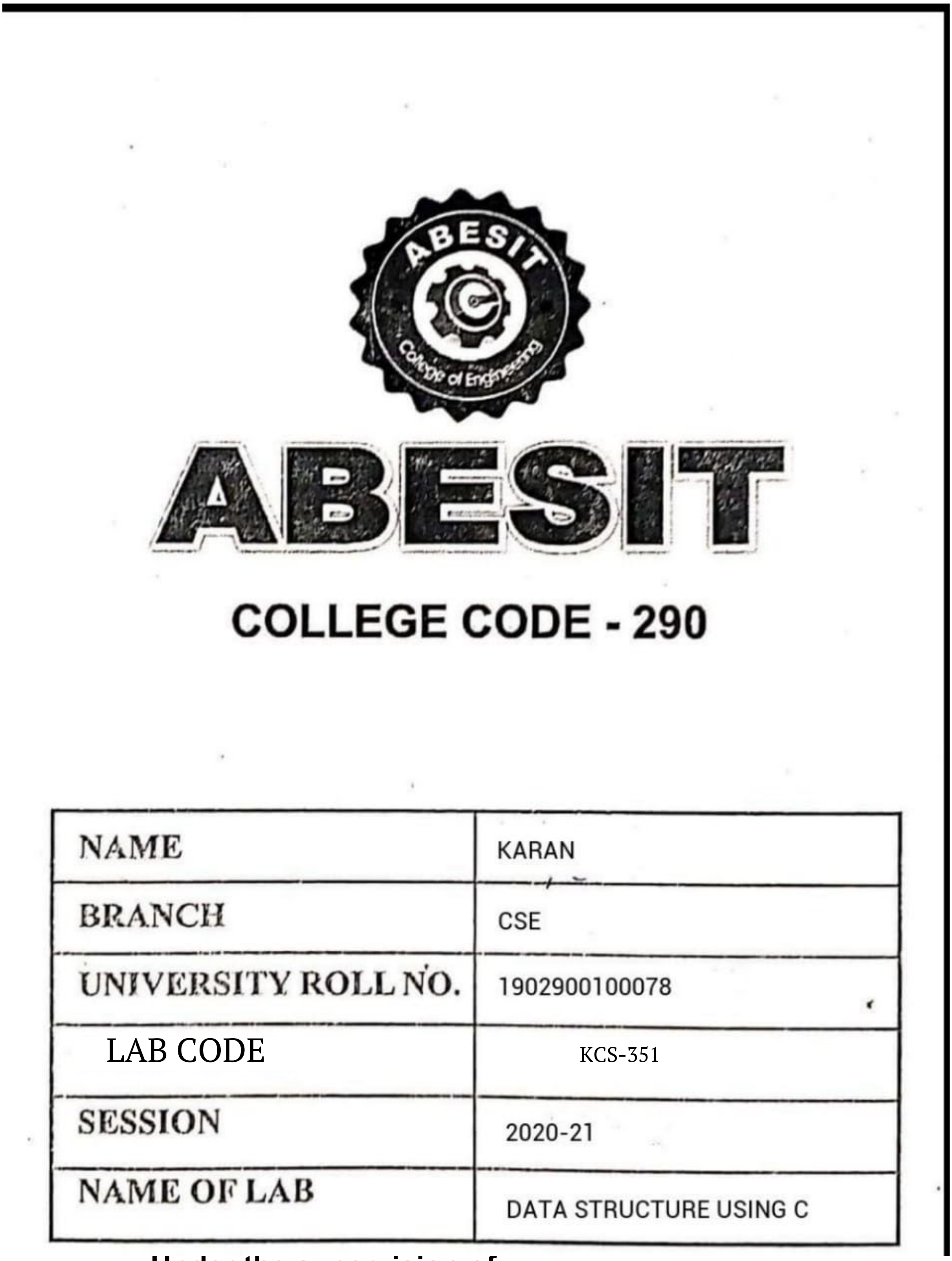


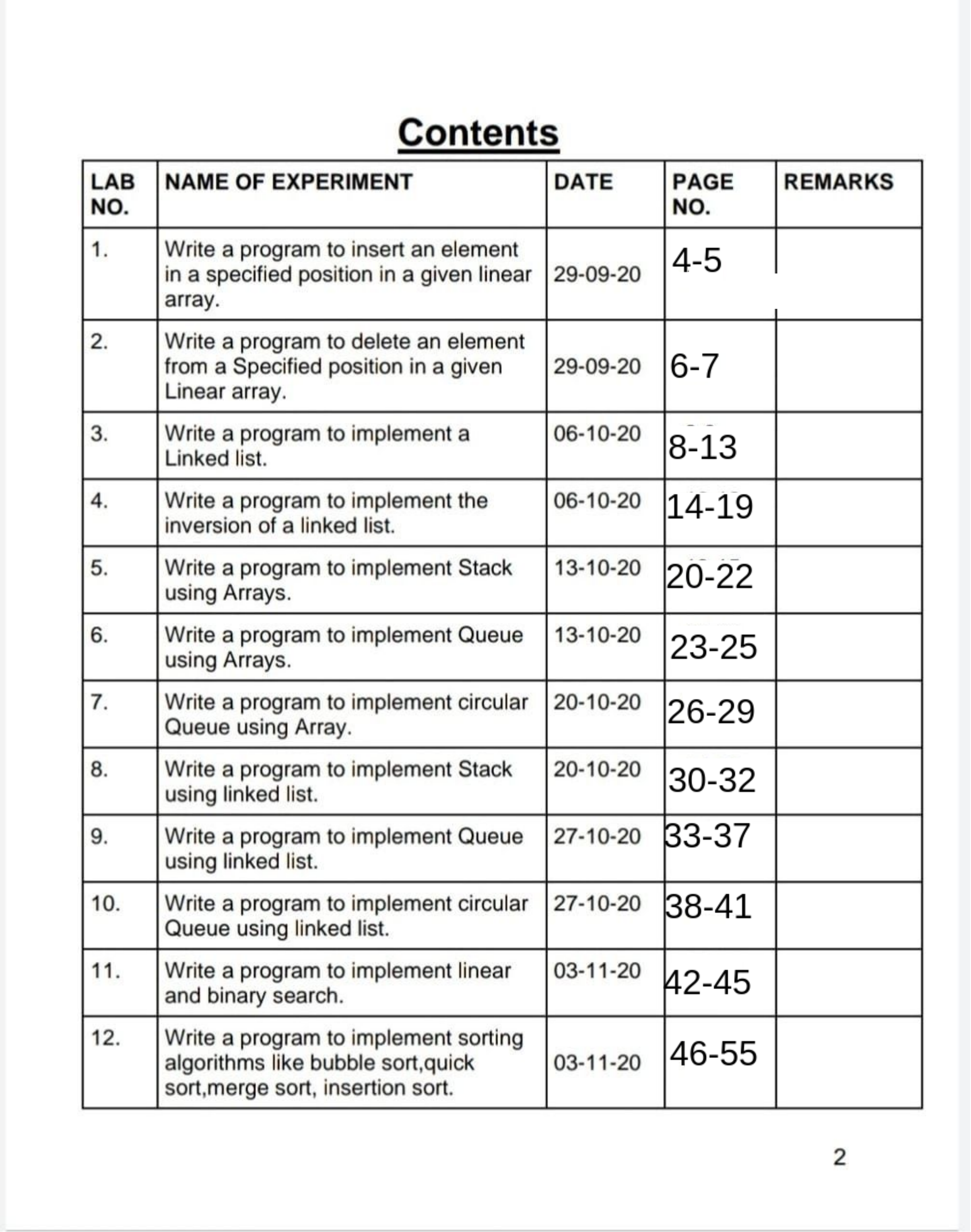
ABESIT

COLLEGE CODE - 290

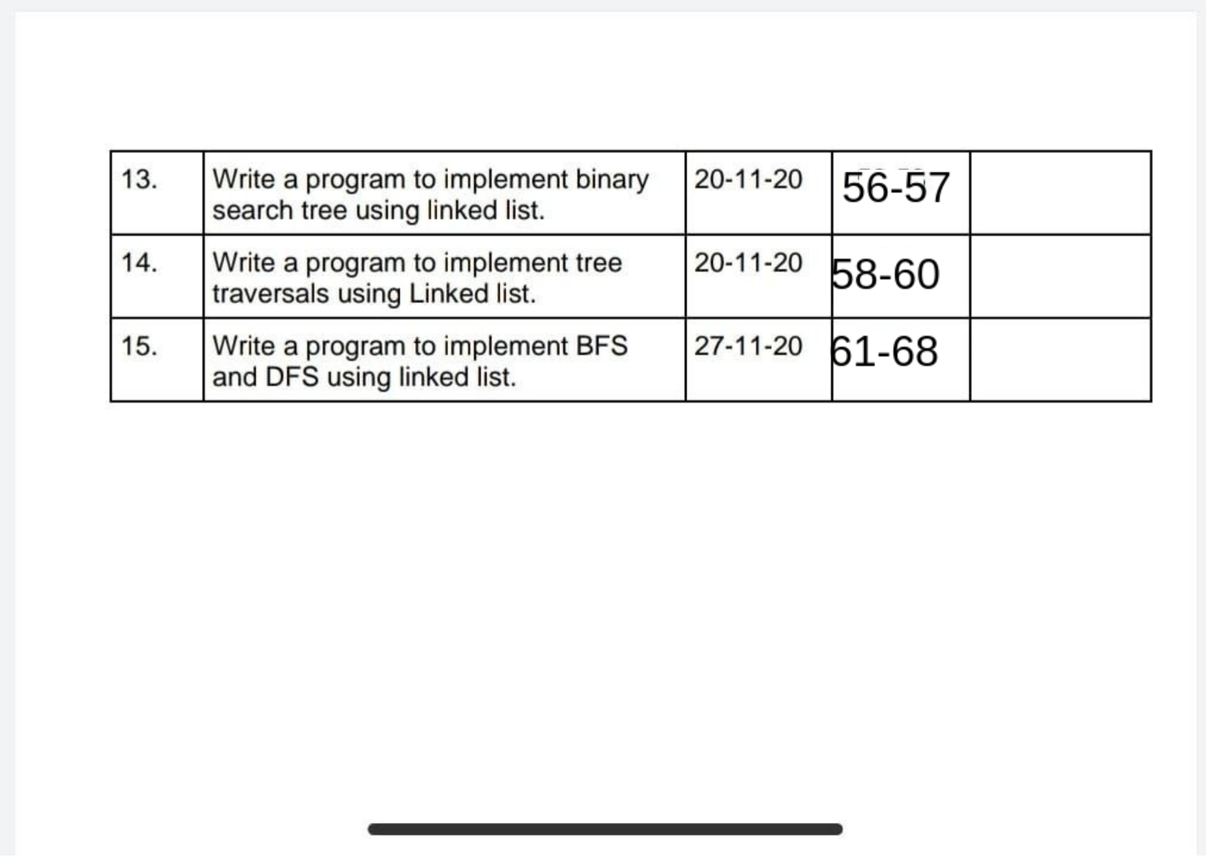
NAME	KARAN
BRANCH	CSE
UNIVERSITY ROLL NO.	1902900100078
LAB CODE	KCS-351
SESSION	2020-21
NAME OF LAB	DATA STRUCTURE USING C

Under the supervision of

Mr. Krishna Bihari Dubey



Contents

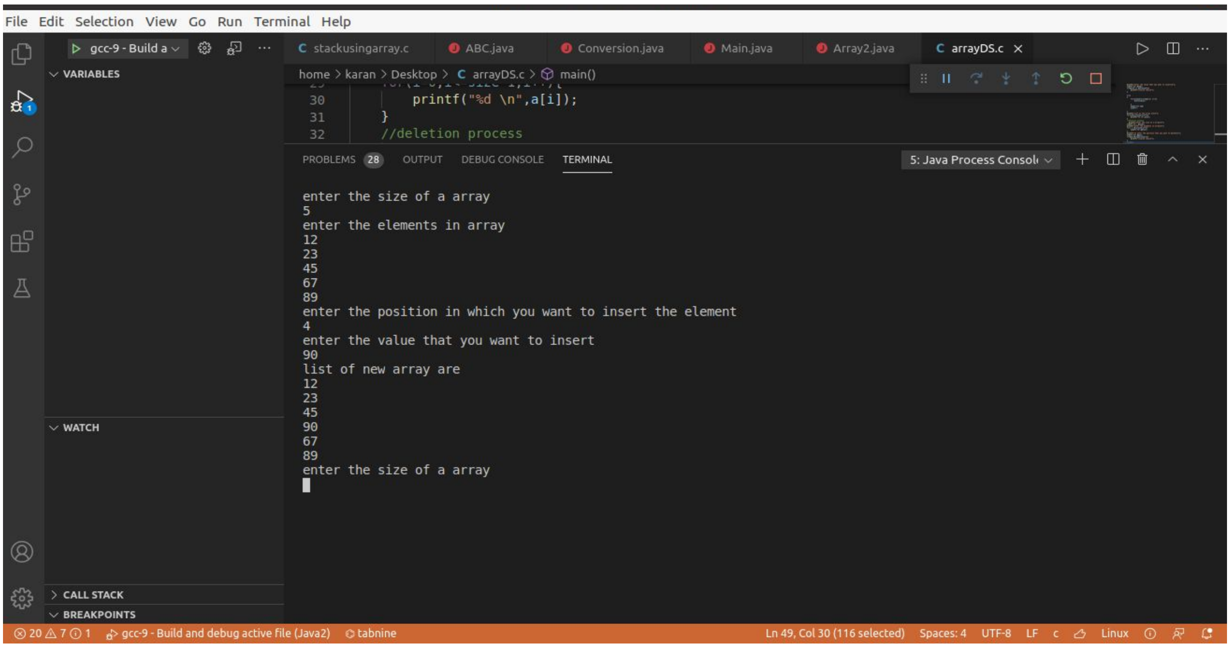


13.	Write a program to implement binary search tree using linked list.	20-11-20	56-57	
14.	Write a program to implement tree traversals using Linked list.	20-11-20	58-60	
15.	Write a program to implement BFS and DFS using linked list.	27-11-20	61-68	

Aim: WAP to insert a give element in a specific position in a given linear array.

Software used: VSC

```
#include<stdio.h>
void main(){
    int a[50],pos,num,size,i;
    printf("enter the size of a array\n");
    scanf("%d", &size);
    printf("enter the elements in array\n");
    for( i =0;i<size;i++){
        scanf("%d",&a[i]);
    }
    printf("enter the position in which you want to insert the 
element\n");
    scanf("%d",&pos);
    printf("enter the value that you want to insert\n");
    scanf("%d",&num);
    if(pos<0 || pos>size+1){
        printf("invaild choice");
    }

    else
    {
        for(i=size-1;i>=pos-1; i--){
            a[i+1]=a[i];

        }
        a[pos-1]= num;
        size++;

    }
    printf("list of new array are\n");
    for(i=0;i<=size-1;i++){
        printf("%d \n",a[i]);
    }
}
```


File Edit Selection View Go Run Terminal Help
gcc-9 - Build a
stackusingarray.c
ABC.java
Conversion.java
Main.java
Array2.java
arrayDS.c
VARIABLES
home > karan > Desktop > arrayDS.c > main()
PROBLEMS 28
OUTPUT
DEBUG CONSOLE
TERMINAL
5: Java Process Console
WATCH
CALL STACK
BREAKPOINTS
20 7 1 gcc-9 - Build and debug active file (Java2) tabnine
Ln 49, Col 30 (116 selected) Spaces: 4 UTF-8 LF c Linux

```
        printf("%d \n",a[i]);
    }
    //deletion process
```

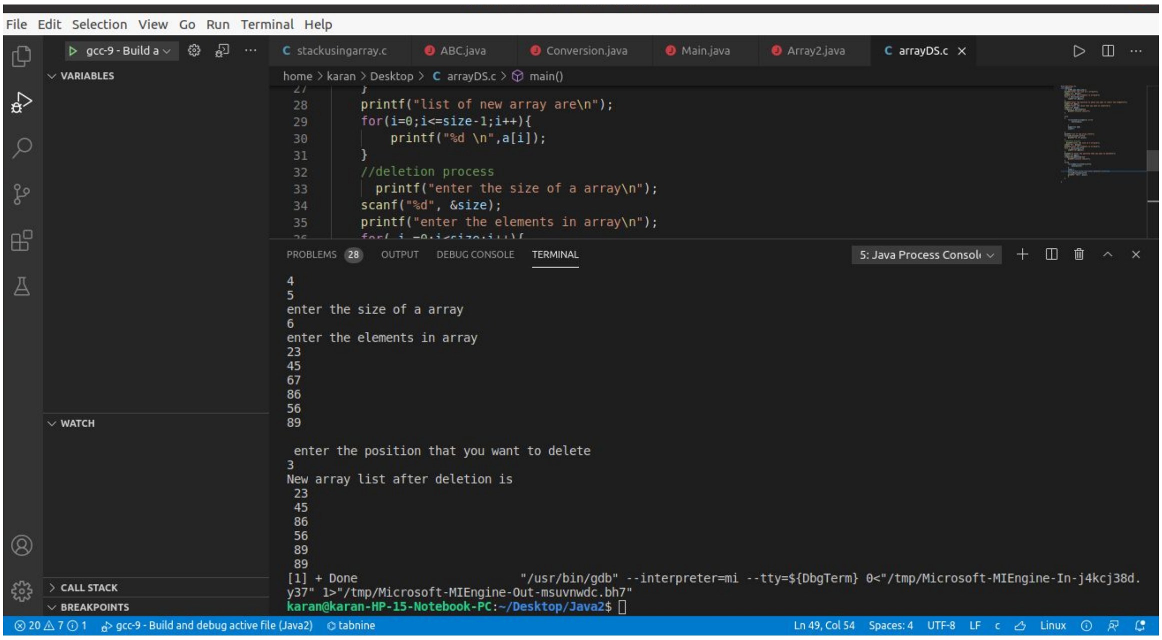
```
enter the size of a array
5
enter the elements in array
12
23
45
67
89
enter the position in which you want to insert the element
4
enter the value that you want to insert
90
list of new array are
12
23
45
90
67
89
enter the size of a array
```

Output:

Aim: WAP to delete a element from a specific position in a given linear array.

Software used: VSC

```
#include<stdio.h>
void main(){
  int a[50],pos,num,size,i;
  printf("enter the size of a array\n");
   scanf("%d", &size);
   printf("enter the elements in array\n");
   for( i =0;i<size;i++){
       scanf("%d",&a[i]);
   }
   printf("\n enter the position that you want to delete\n");
   scanf("%d",&pos);
   if(pos<=0||pos>size-1){
       printf("invaild choice");
   }
   else{
       for(i=pos-1;i<size-1;i++){
           a[i]=a[i+1];
       }
       size--;
       for(i=0;i<=size;i++){
       printf("new array is %d\n" ,a[i]);
       }
   }


}
```



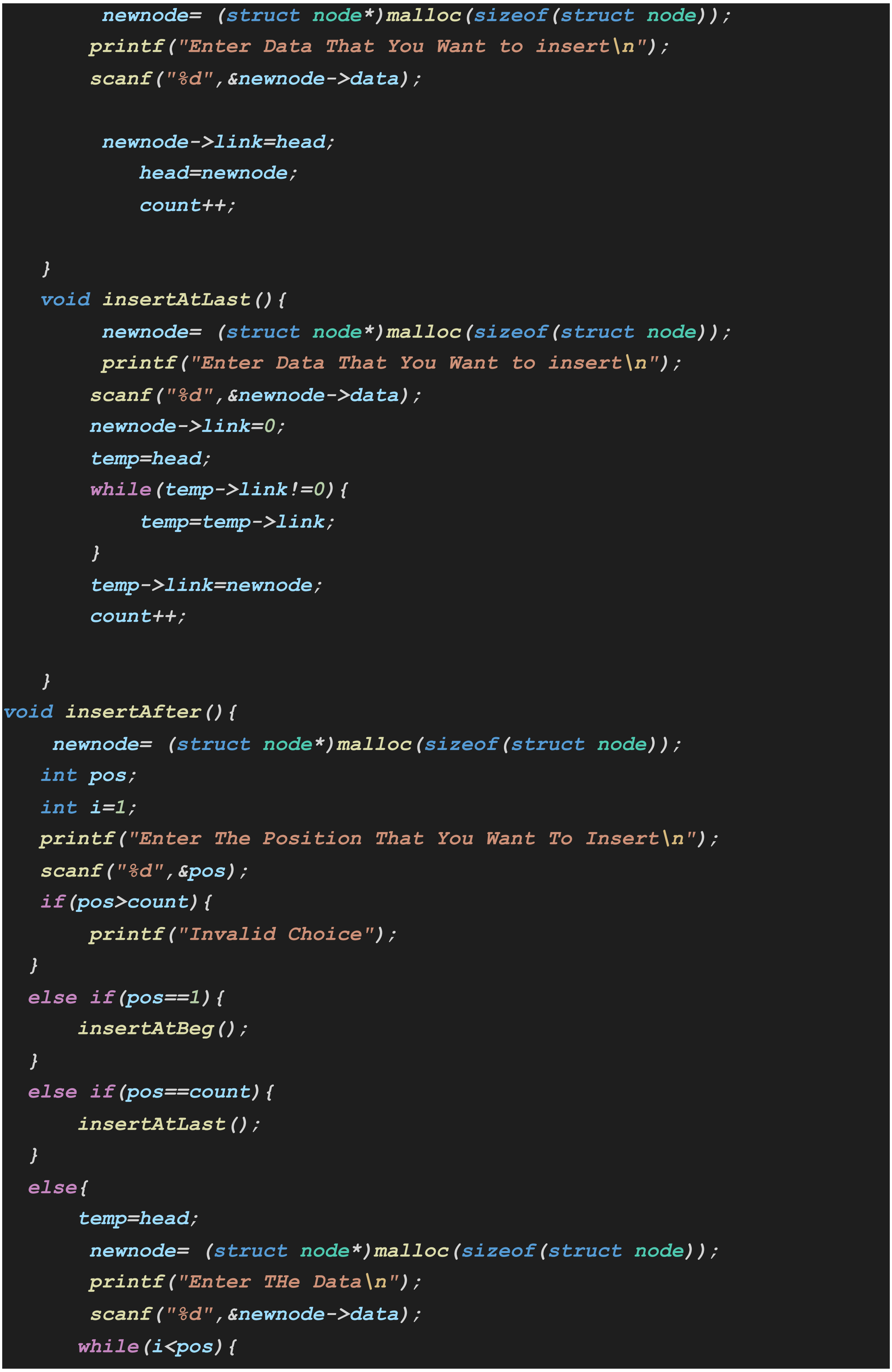
```c
    }
    printf("list of new array are\n");
    for(i=0;i<=size-1;i++){
        printf("%d \n",a[i]);
    }
    //deletion process
     printf("enter the size of a array\n");
    scanf("%d", &size);
    printf("enter the elements in array\n");
    for( i =0;i<size;i++){
```

```
4
5
enter the size of a array
6
enter the elements in array
23
45
67
86
56
89

 enter the position that you want to delete
3
New array list after deletion is
 23
 45
 86
 56
 89
 89
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-j4kcj38d.
y37" 1>"/tmp/Microsoft-MIEngine-Out-msuvnwdc.bh7"
karan@karan-HP-15-Notebook-PC:~/Desktop/Java2$ 
```

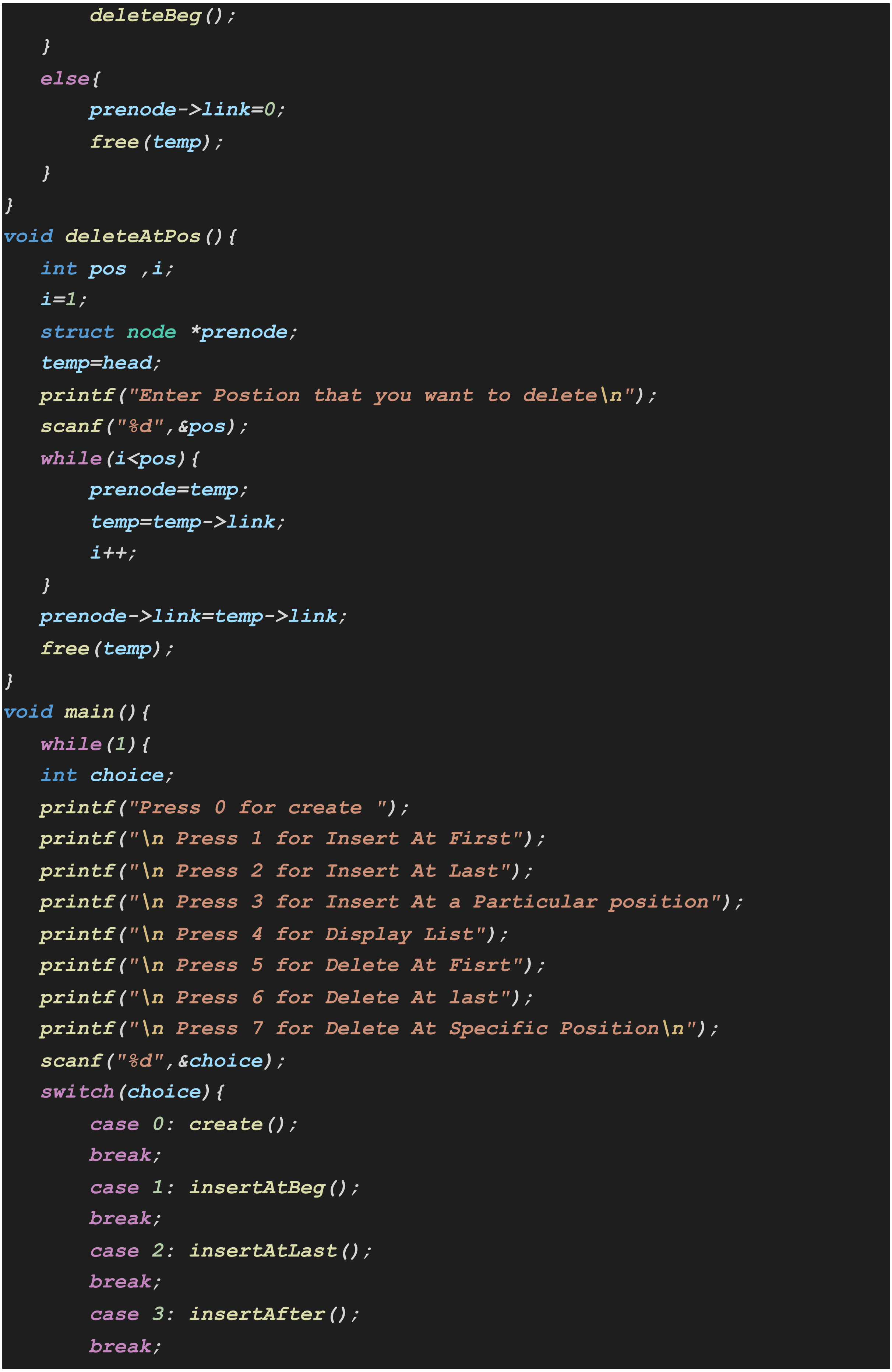
output:

Aim: WAP to implement a linked list.
Software used:VSC

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


struct node{
   int data;
   struct node *link;

};
struct node *head;


int count=0;

struct node *newnode,*temp;
void create(){
   head=0;
   int choice;
   while(choice){
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("Enter the data\n");
   scanf("%d",&newnode->data);
   newnode->link=0;
   if(head==0){
       head=newnode;
   }
   else{
       struct node *temp;
       temp=head;
       temp->link=newnode;
       temp=newnode;

   }
   printf("If you want to add more then press 1\n or else press
zero(0)");
   scanf("%d",&choice);
   count++;
}

}
   void insertAtBeg(){
```

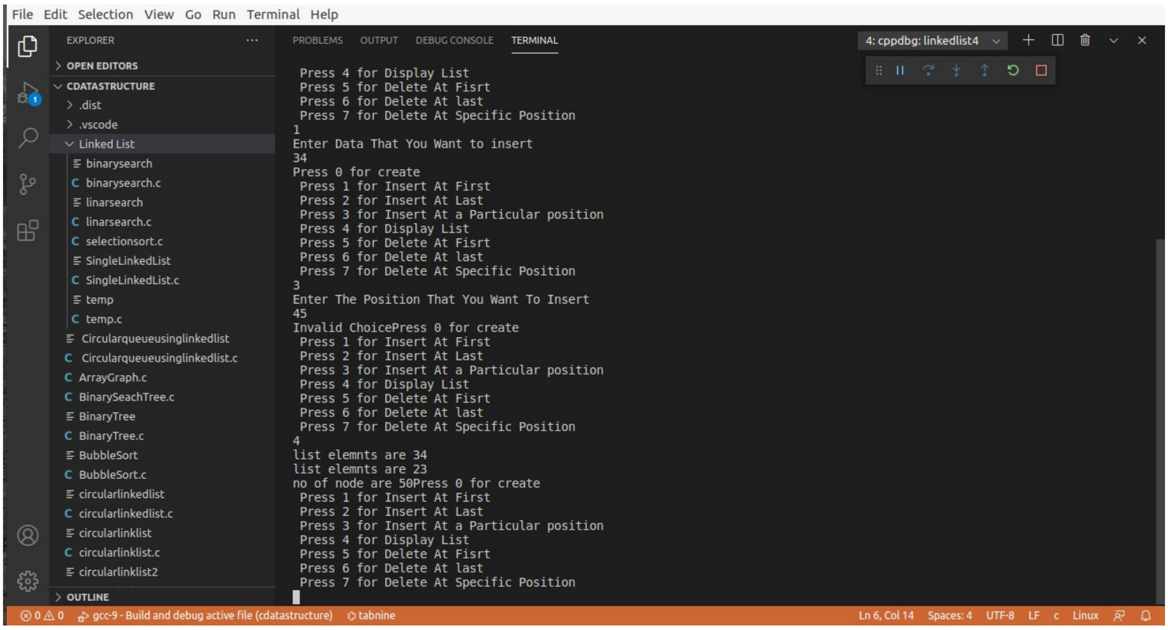


```
        newnode= (struct node*)malloc(sizeof(struct node));
       printf("Enter Data That You Want to insert\n");
       scanf("%d",&newnode->data);

        newnode->link=head;
           head=newnode;
           count++;

   }
   void insertAtLast(){
        newnode= (struct node*)malloc(sizeof(struct node));
        printf("Enter Data That You Want to insert\n");
       scanf("%d",&newnode->data);
       newnode->link=0;
       temp=head;
       while(temp->link!=0){
           temp=temp->link;
       }
       temp->link=newnode;
       count++;

   }
void insertAfter(){
    newnode= (struct node*)malloc(sizeof(struct node));
   int pos;
   int i=1;
   printf("Enter The Position That You Want To Insert\n");
   scanf("%d",&pos);
   if(pos>count){
       printf("Invalid Choice");
  }
  else if(pos==1){
      insertAtBeg();
  }
  else if(pos==count){
      insertAtLast();
  }
  else{
      temp=head;
       newnode= (struct node*)malloc(sizeof(struct node));
       printf("Enter THe Data\n");
       scanf("%d",&newnode->data);
      while(i<pos){
          temp=temp->link;
          i++;
      }
       newnode->link=temp->link;
       temp->link=newnode;
  }
  count++;
}

void display(){
   if(head==0){
       printf("list is empty");
   }
   else{
       temp=head;
       while(temp!=0){
           printf("list elemnts are %d \n", temp->data);
           temp=temp->link;
           count++;
       }
       printf("no of node are %d0",count);

}
}
void deleteBeg(){
   if(head==0){
       printf("list is empty\n");
   }
   else{
   temp=head;
   head=head->link;
   free(temp);
}
}
void deleteEnd(){
   struct node *prenode;
   temp=head;
   while(temp->link!=0){
       prenode=temp;
       temp=temp->link;

   }
   if(head==temp){
```



```
        deleteBeg();
    }
    else{
        prenode->link=0;
        free(temp);
    }
}
void deleteAtPos(){
    int pos ,i;
    i=1;
    struct node *prenode;
    temp=head;
    printf("Enter Postion that you want to delete\n");
    scanf("%d",&pos);
    while(i<pos){
        prenode=temp;
        temp=temp->link;
        i++;
    }
    prenode->link=temp->link;
    free(temp);
}
void main(){
    while(1){
    int choice;
    printf("Press 0 for create ");
    printf("\n Press 1 for Insert At First");
    printf("\n Press 2 for Insert At Last");
    printf("\n Press 3 for Insert At a Particular position");
    printf("\n Press 4 for Display List");
    printf("\n Press 5 for Delete At Fisrt");
    printf("\n Press 6 for Delete At last");
    printf("\n Press 7 for Delete At Specific Position\n");
    scanf("%d",&choice);
    switch(choice){
        case 0: create();
        break;
        case 1: insertAtBeg();
        break;
        case 2: insertAtLast();
        break;
        case 3: insertAfter();
        break;
```

```
 case 4: display();
 break;
 case 5: deleteBeg();
 break;
 case 6: deleteEnd();
 break;
 case 7: deleteAtPos();
 break;
  default:
     printf("wrong choice");
         break;
     }
}
     }
```

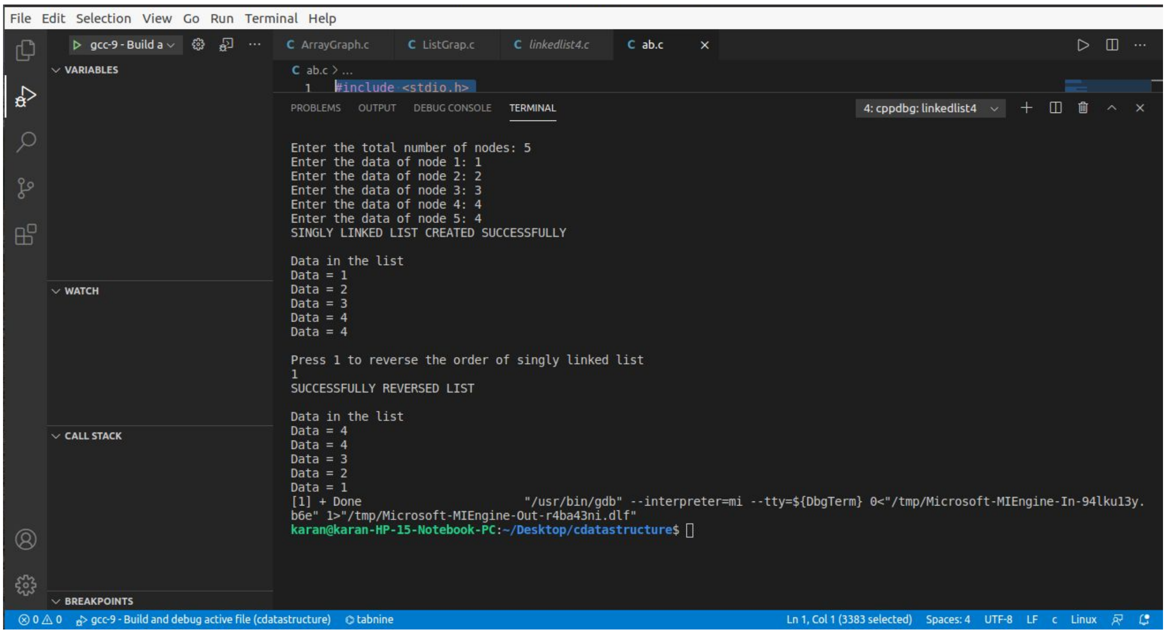


```
 Press 4 for Display List
 Press 5 for Delete At Fisrt
 Press 6 for Delete At last
 Press 7 for Delete At Specific Position
1
Enter Data That You Want to insert
34
Press 0 for create
 Press 1 for Insert At First
 Press 2 for Insert At Last
 Press 3 for Insert At a Particular position
 Press 4 for Display List
 Press 5 for Delete At Fisrt
 Press 6 for Delete At last
 Press 7 for Delete At Specific Position
3
Enter The Position That You Want To Insert
45
Invalid ChoicePress 0 for create
 Press 1 for Insert At First
 Press 2 for Insert At Last
 Press 3 for Insert At a Particular position
 Press 4 for Display List
 Press 5 for Delete At Fisrt
 Press 6 for Delete At last
 Press 7 for Delete At Specific Position
4
list elemnts are 34
list elemnts are 23
no of node are 50Press 0 for create
 Press 1 for Insert At First
 Press 2 for Insert At Last
 Press 3 for Insert At a Particular position
 Press 4 for Display List
 Press 5 for Delete At Fisrt
 Press 6 for Delete At last
 Press 7 for Delete At Specific Position
```

Aim: WAP to reverse a linked list

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


/* Structure of a node */
struct node {
    int data; //Data part
    struct node *next; //Address part
}*head;


/* Functions used in the program */
void createList(int n);
void reverseList();
void displayList();


int main()
{
    int n, choice;

    /*
     * Create a singly linked list of n nodes
     */
    printf("Enter the total number of nodes: ");
    scanf("%d", &n);
    createList(n);

    printf("\nData in the list \n");
    displayList();

    /*
     * Reverse the list
     */
    printf("\nPress 1 to reverse the order of singly linked list\n");
    scanf("%d", &choice);
    if(choice == 1)
    {
        reverseList();
    }

    printf("\nData in the list\n");
    displayList();

    return 0;
}



/*
* Create a list of n nodes
*/
void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;

    if(n <= 0)
    {
        printf("List size must be greater than zero.\n");
        return;
    }

    head = (struct node *)malloc(sizeof(struct node));

    /*
     * If unable to allocate memory for head node
     */
    if(head == NULL)
    {
        printf("Unable to allocate memory.");
    }
    else
    {
        /*
         * Read data of node from the user
         */
        printf("Enter the data of node 1: ");
        scanf("%d", &data);

        head->data = data; // Link the data field with data
        head->next = NULL; // Link the address field to NULL

        temp = head;

        /*
```

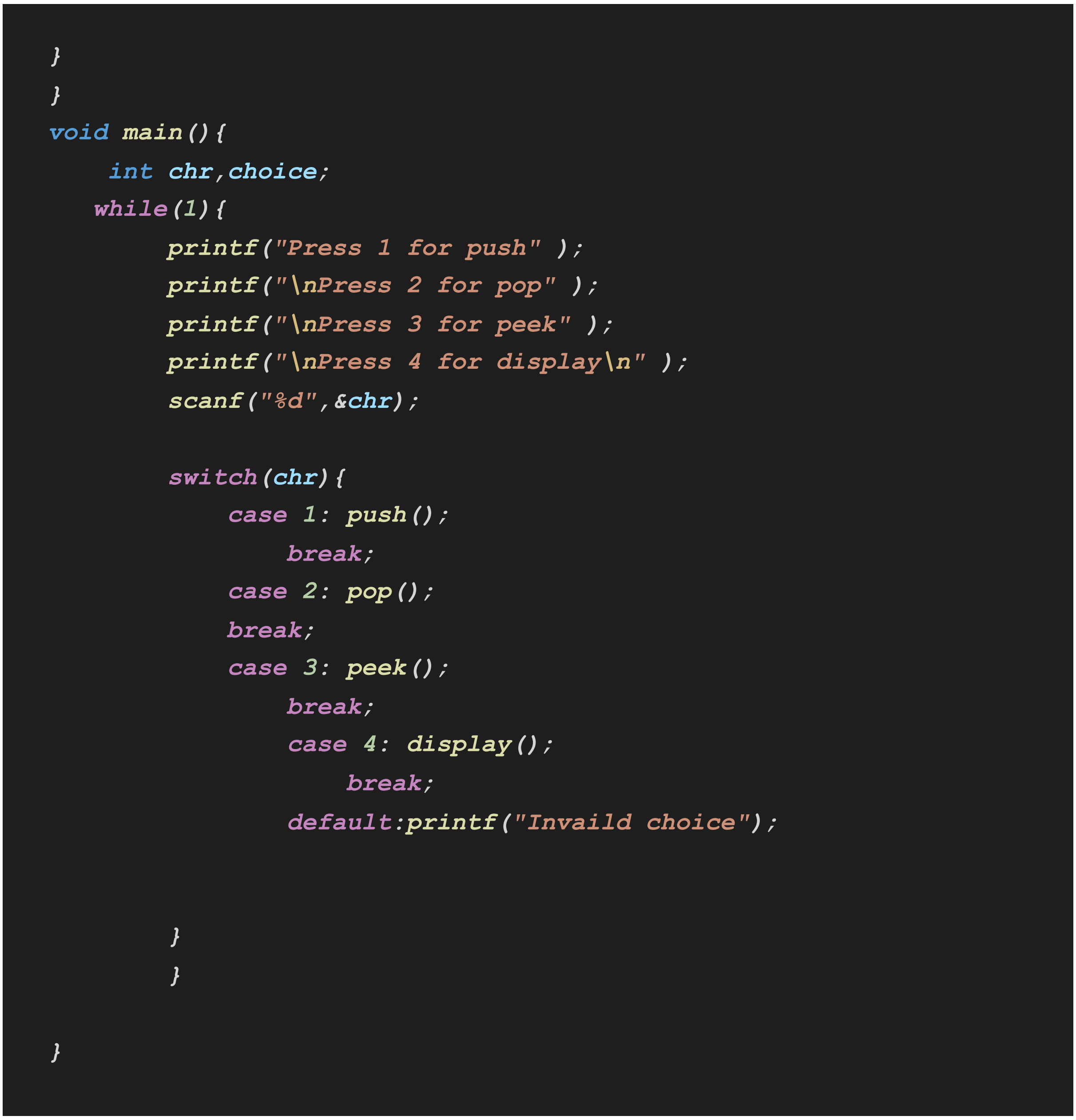
```
         * Create n nodes and adds to linked list
         */
        for(i=2; i<=n; i++)
        {
            newNode = (struct node *)malloc(sizeof(struct node));

            /* If memory is not allocated for newNode */
            if(newNode == NULL)
            {
                printf("Unable to allocate memory.");
                break;
            }
            else
            {
                printf("Enter the data of node %d: ", i);
                scanf("%d", &data);

                newNode->data = data; // Link the data field of newNode
with data
                newNode->next = NULL; // Link the address field of
newNode with NULL

                temp->next = newNode; // Link previous node i.e. temp to
the newNode
                temp = temp->next;
            }
        }

        printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");
    }
}


/*
* Reverse the order of nodes of a singly linked list
*/
void reverseList()
{
    struct node *prevNode, *curNode;

    if(head != NULL)
    {
        prevNode = head;
        curNode = head->next;
        head = head->next;

        prevNode->next = NULL; // Make first node as last node

        while(head != NULL)
        {
            head = head->next;
            curNode->next = prevNode;

            prevNode = curNode;
            curNode = head;
        }

        head = prevNode; // Make last node as head

        printf("SUCCESSFULLY REVERSED LIST\n");
    }
}


/*
* Display entire list
*/
void displayList()
{
    struct node *temp;

    /*
     * If the list is empty i.e. head = NULL
     */
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        temp = head;
        while(temp != NULL)
        {
            printf("Data = %d\n", temp->data); // Print the data of
current node
            temp = temp->next;                 // Move to next node
        }
    }
}
```



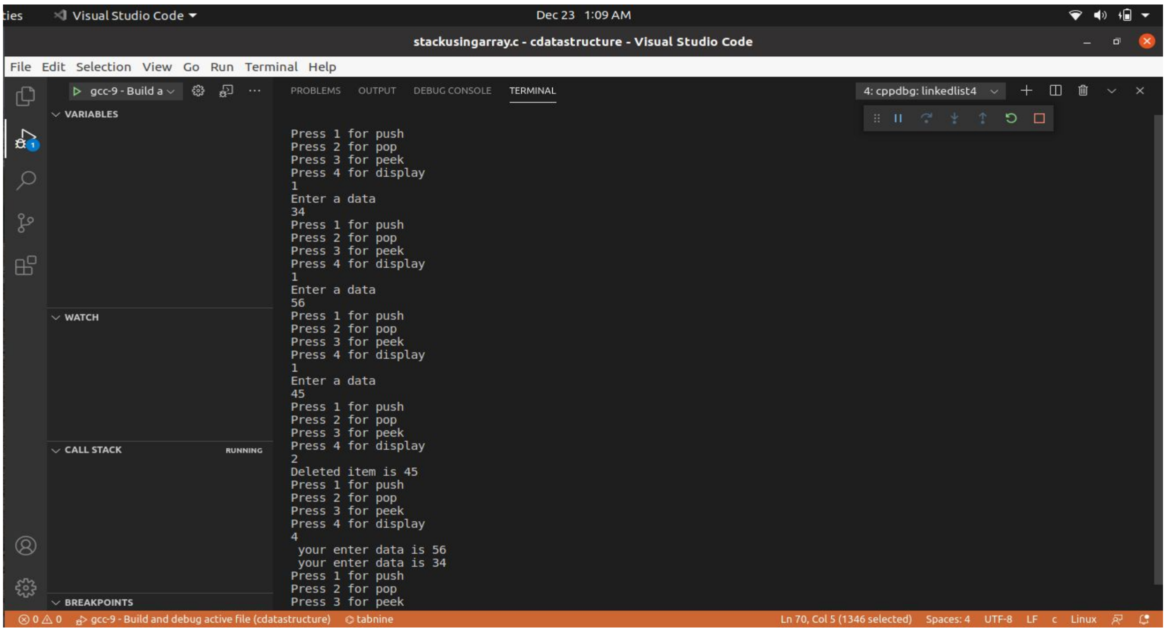
```
Enter the total number of nodes: 5
Enter the data of node 1: 1
Enter the data of node 2: 2
Enter the data of node 3: 3
Enter the data of node 4: 4
Enter the data of node 5: 4
SINGLY LINKED LIST CREATED SUCCESSFULLY

Data in the list
Data = 1
Data = 2
Data = 3
Data = 4
Data = 4

Press 1 to reverse the order of singly linked list
1
SUCCESSFULLY REVERSED LIST

Data in the list
Data = 4
Data = 4
Data = 3
Data = 2
Data = 1
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-94lku13y.
b6e" 1>"/tmp/Microsoft-MIEngine-Out-r4ba43ni.dlf"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

Aim: WAP to implement a stack usin array.
Software used: VSC

```
#include<stdio.h>

int stack[5];
int top=-1;
void push(){
   int x;
   printf("Enter a data\n");
   scanf("%d",&x);
   if(top==5-1){
       printf("over flow\n");

   }
   else{
       top++;
       stack[top]=x;
   }
}
void pop(){
   int item;
   if(top==-1){
       printf("under flow\n");
   }
   else{
       item=stack[top];
       top--;
       printf("Deleted item is %d\n",item);
   }


}
void peek(){
   if(top==-1){
       printf("stack is empty\n");
   }
   else{
   printf("top element in stact is  %d\n", stack[top]);
   }
}
   void display(){
       int i;
       for(i=top;i>=0;i--){
           printf(" your enter data is %d\n", stack[i]);
```



```
}
}
void main(){
    int chr,choice;
   while(1){
        printf("Press 1 for push" );
        printf("\nPress 2 for pop" );
        printf("\nPress 3 for peek" );
        printf("\nPress 4 for display\n" );
        scanf("%d",&chr);

        switch(chr){
            case 1: push();
                break;
            case 2: pop();
            break;
            case 3: peek();
                break;
                case 4: display();
                    break;
                default:printf("Invaild choice");


        }
        }

}
```

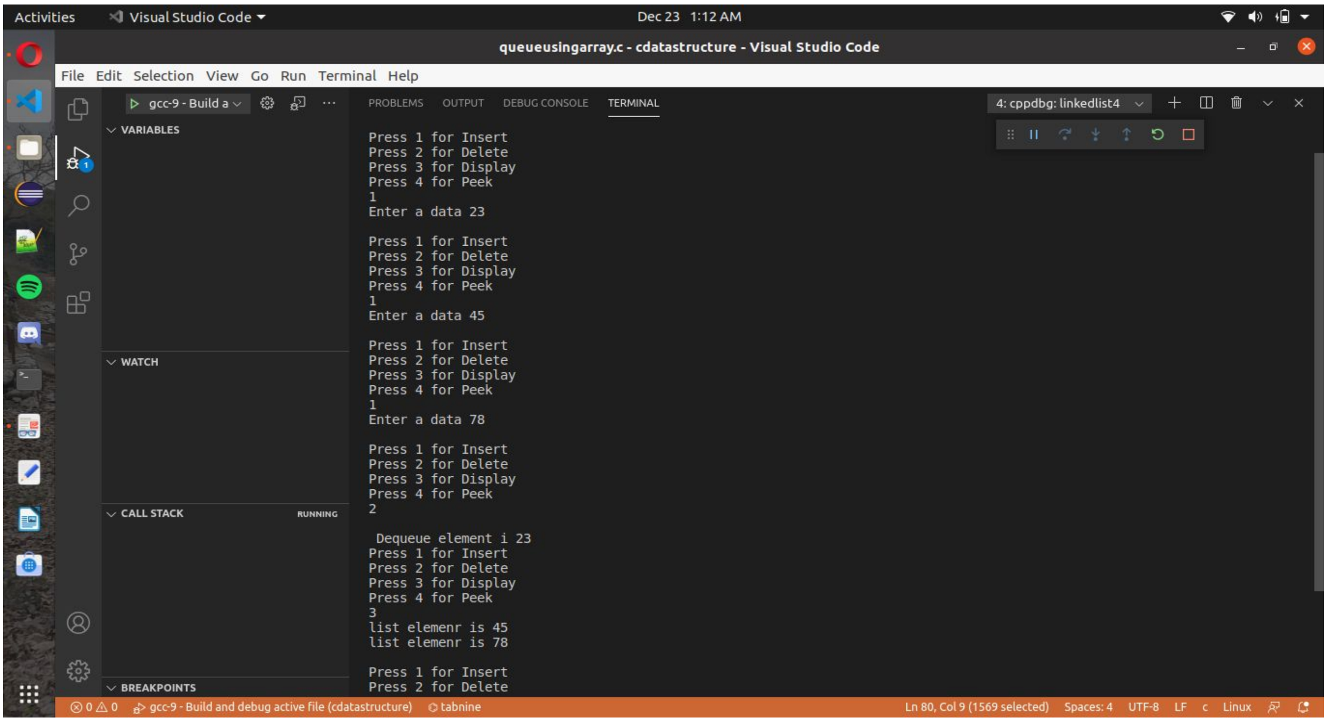


```
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
1
Enter a data
34
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
1
Enter a data
56
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
1
Enter a data
45
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
2
Deleted item is 45
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
4
 your enter data is 56
 your enter data is 34
Press 1 for push
Press 2 for pop
Press 3 for peek
```

Aim: WAP to implement queue using array.
Software used: VSC

```
#include<stdio.h>
int queue[5];
int front =-1;
int rear=-1;
void enqeue(){
    int x;
     printf("Enter a data ");
    scanf("%d",&x);
    if(rear==5-1){
        printf("overflow");
    }
    else if(rear==-1&&front==-1){
        front=rear=0;
        queue[rear]=x;
    }
    else{
        rear++;
        queue[rear]=x;
    }
}
void dequeue(){
    if(front==-1&&rear==-1){
        printf("overflow");
    }
    else if(front==rear){
        front=rear=-1;
    }
    else{
        printf("\n Dequeue element i %d ",queue[front]);
        front++;
    }

}
void display(){
    if(front==-1&&rear==-1){
        printf("underflow");

    }
    else{
        for(int i=front;i<rear+1;i++){
            printf("list elemenr is %d\n",queue[i]);
```

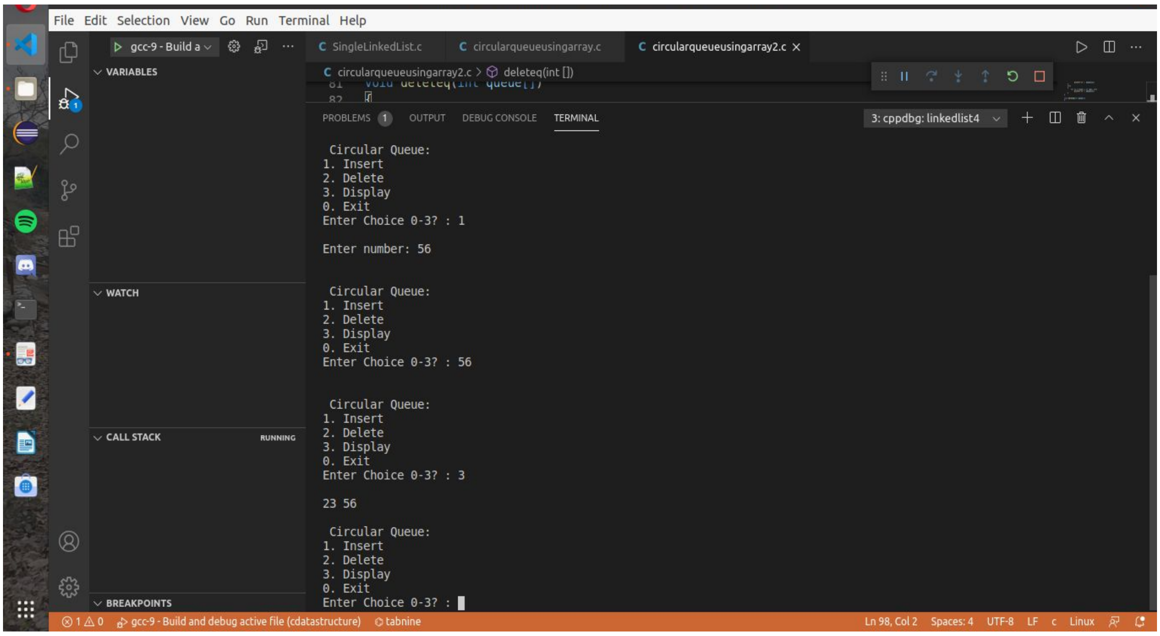
```
       }
   }
}
void peek(){
   if(front==-1&&rear==-1){
       printf("over flow ");
   }
   else{
       printf("1st element is %d ",queue[front]);
   }
}
void main(){
       int chr;
      while(chr){
           printf("\nPress 1 for Insert" );
           printf("\nPress 2 for Delete" );
           printf("\nPress 3 for Display" );
           printf("\nPress 4 for Peek\n" );

           scanf("\n%d",&chr);
           switch (chr)
           {
           case 1: enqeue();
               break;

           case 2: dequeue();
               break;
           case 3: display();
               break;
           case 4: peek();
           break;

           default:
           printf("wrong choice");
               break;
           }
      }
}
```



```
Press 1 for Insert
Press 2 for Delete
Press 3 for Display
Press 4 for Peek
1
Enter a data 23

Press 1 for Insert
Press 2 for Delete
Press 3 for Display
Press 4 for Peek
1
Enter a data 45

Press 1 for Insert
Press 2 for Delete
Press 3 for Display
Press 4 for Peek
1
Enter a data 78

Press 1 for Insert
Press 2 for Delete
Press 3 for Display
Press 4 for Peek
2

 Dequeue element i 23
Press 1 for Insert
Press 2 for Delete
Press 3 for Display
Press 4 for Peek
3
list elemenr is 45
list elemenr is 78

Press 1 for Insert
Press 2 for Delete
```

Aim : WAP to implement circular queue using array.

```
#include <stdio.h>
#define size 5

void insertq(int[], int);
void deleteq(int[]);
void display(int[]);

int front =  - 1;
int rear =  - 1;

int main()
{
   int n, ch;
   int queue[size];
   do
   {
       printf("\n\n Circular Queue:\n1. Insert \n2. Delete\n3.
Display\n0. Exit");
       printf("\nEnter Choice 0-3? : ");
       scanf("%d", &ch);
       switch (ch)
       {
           case 1:
               printf("\nEnter number: ");
               scanf("%d", &n);
               insertq(queue, n);
               break;
           case 2:
               deleteq(queue);
               break;
           case 3:
               display(queue);
               break;
       }
   }while (ch != 0);
}


void insertq(int queue[], int item)
{
   if ((front == 0 && rear == size - 1) || (front == rear + 1))
   {
       printf("queue is full");
       return;
   }
   else if (rear ==  - 1)
   {
       rear++;
       front++;
   }
   else if (rear == size - 1 && front > 0)
   {
       rear = 0;
   }
   else
   {
       rear++;
   }
   queue[rear] = item;
}

void display(int queue[])
{
   int i;
   printf("\n");
   if (front > rear)
   {
       for (i = front; i < size; i++)
       {
           printf("%d ", queue[i]);
       }
       for (i = 0; i <= rear; i++)
           printf("%d ", queue[i]);
   }
   else
   {
       for (i = front; i <= rear; i++)
           printf("%d ", queue[i]);
   }
}

void deleteq(int queue[])
{
   if (front ==  - 1)
   {
       printf("Queue is empty ");
   }
   else if (front == rear)
   {
       printf("\n %d deleted", queue[front]);
       front =  - 1;
       rear =  - 1;
   }
   else
   {
       printf("\n %d deleted", queue[front]);
       front++;
   }
}
```

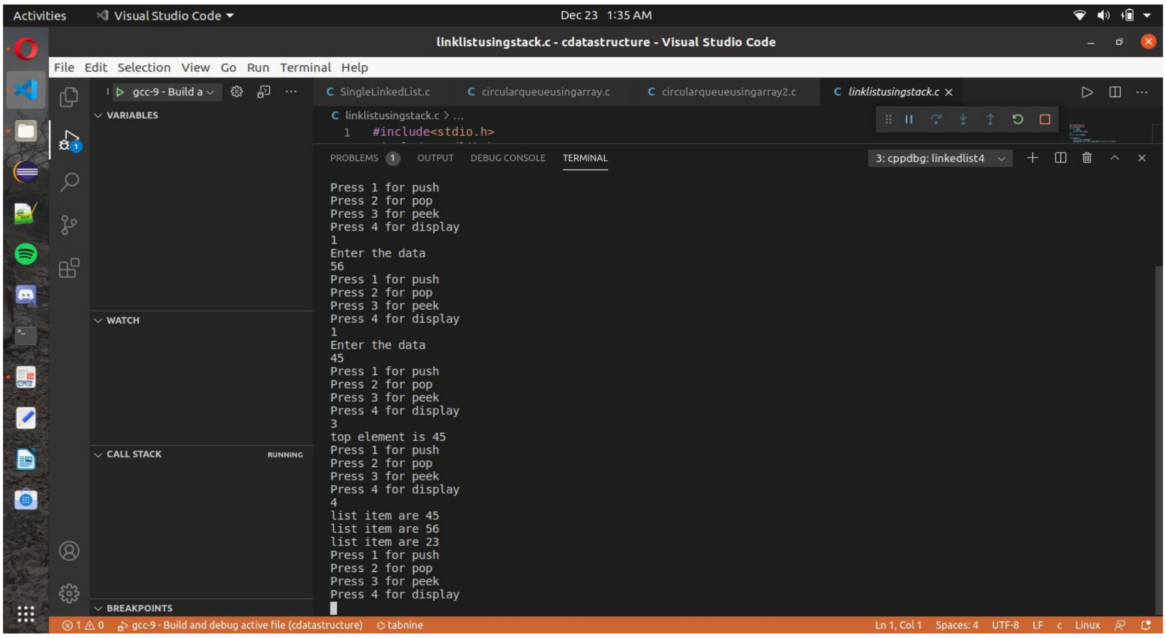

File Edit Selection View Go Run Terminal Help
gcc-9 - Build a
SingleLinkedList.c
circularqueueusingarray.c
circularqueueusingarray2.c
circularqueueusingarray2.c > deleteq(int [])
VARIABLES
WATCH
CALL STACK
RUNNING
BREAKPOINTS
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
3: cppdbg: linkedlist4
⊗ 1 ⚠ 0
gcc-9 - Build and debug active file (cdatastructure)
tabnine
Ln 98, Col 2
Spaces: 4
UTF-8
LF
c
Linux

```
 Circular Queue:
1. Insert
2. Delete
3. Display
0. Exit
Enter Choice 0-3? : 1

Enter number: 56


 Circular Queue:
1. Insert
2. Delete
3. Display
0. Exit
Enter Choice 0-3? : 56


 Circular Queue:
1. Insert
2. Delete
3. Display
0. Exit
Enter Choice 0-3? : 3

23 56

 Circular Queue:
1. Insert
2. Delete
3. Display
0. Exit
Enter Choice 0-3? :
```

Aim: WAP to implement stack using linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node *next;
};
struct node *top=0;

void push(){
    struct node *newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
    printf("Enter the data\n");
    scanf("%d",&newnode->data);
    newnode->next=0;
    if(top==0){
        top=newnode;
    }
    else{
        newnode->next=top;
          top=newnode;
    }

}
void pop(){
    struct node *temp;
    temp=top;
    if(top==0){
        printf("Stack is empty\n");
    }
    else{
        printf("pop element is %d\n",top->data);
        top=top->next;
        free(temp);
    }

}
void peek(){
    if(top==0){
        printf("stack is empty\n");
    }
    else{
        printf("top element is %d\n",top->data);
```

```
   }
}
void display(){
struct node *temp;
temp=top;
if(temp==0){
   printf("stack is empty\n");
}
   else{
       while(temp!=0){
           printf("list item are %d\n",temp->data);
           temp=temp->next;
       }
   }
}
void main(){
       int chr,choice;
      while(1){
           printf("Press 1 for push" );
           printf("\nPress 2 for pop" );
           printf("\nPress 3 for peek" );
           printf("\nPress 4 for display\n" );
           scanf("%d",&chr);

           switch(chr){
               case 1: push();
                   break;
               case 2: pop();
               break;
               case 3: peek();
                   break;
                   case 4: display();
                       break;
                   default:printf("Invaild choice");


           }
           }


   }
```



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

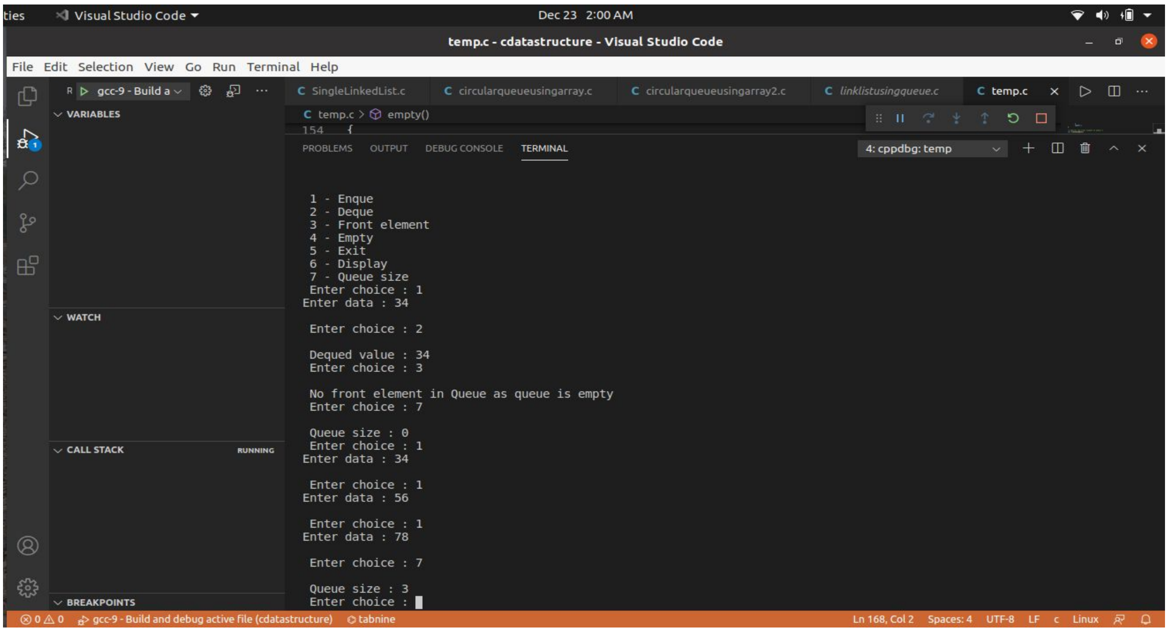
```
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
1
Enter the data
56
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
1
Enter the data
45
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
3
top element is 45
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
4
list item are 45
list item are 56
list item are 23
Press 1 for push
Press 2 for pop
Press 3 for peek
Press 4 for display
```

Aim: WAP to implement queue using linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *ptr;
}*front,*rear,*temp,*front1;
int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();
int count = 0;
void main()
{
   int no, ch, e;
   printf("\n 1 - Enque");
   printf("\n 2 - Deque");
   printf("\n 3 - Front element");
   printf("\n 4 - Empty");
   printf("\n 5 - Exit");
   printf("\n 6 - Display");
   printf("\n 7 - Queue size");
   create();
   while (1)
   {
       printf("\n Enter choice : ");
       scanf("%d", &ch);
       switch (ch)
       {
       case 1:
           printf("Enter data : ");
           scanf("%d", &no);
           enq(no);
           break;
       case 2:
           deq();
           break;
       case 3:
           e = frontelement();
```

```
            if (e != 0)
                printf("Front element : %d", e);
            else
                printf("\n No front element in Queue as queue is
empty");
            break;
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            queuesize();
            break;
        default:
            printf("Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}
/* Create an empty queue */
void create()
{
    front = rear = NULL;
}
/* Returns queue size */
void queuesize()
{
    printf("\n Queue size : %d", count);
}
/* Enqueing the queue */
void enq(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }
```

```
   else
   {
       temp=(struct node *)malloc(1*sizeof(struct node));
       rear->ptr = temp;
       temp->info = data;
       temp->ptr = NULL;
       rear = temp;
   }
   count++;
}
/* Displaying the queue elements */
void display()
{
   front1 = front;
   if ((front1 == NULL) && (rear == NULL))
   {
       printf("Queue is empty");
       return;
   }
   while (front1 != rear)
   {
       printf("%d ", front1->info);
       front1 = front1->ptr;
   }
   if (front1 == rear)
       printf("%d", front1->info);
}
/* Dequeing the queue */
void deq()
{
   front1 = front;
   if (front1 == NULL)
   {
       printf("\n Error: Trying to display elements from empty queue");
       return;
   }
   else
       if (front1->ptr != NULL)
       {
           front1 = front1->ptr;
           printf("\n Dequed value : %d", front->info);
           free(front);
           front = front1;
       }
       else
       {
           printf("\n Dequed value : %d", front->info);
           free(front);
           front = NULL;
           rear = NULL;
       }
       count--;
}
/* Returns the front element of queue */
int frontelement()
{
   if ((front != NULL) && (rear != NULL))
       return(front->info);
   else
       return 0;
}
/* Display if queue is empty or not */
void empty()
{
    if ((front == NULL) && (rear == NULL))
       printf("\n Queue empty");
   else
      printf("Queue not empty");
}
```



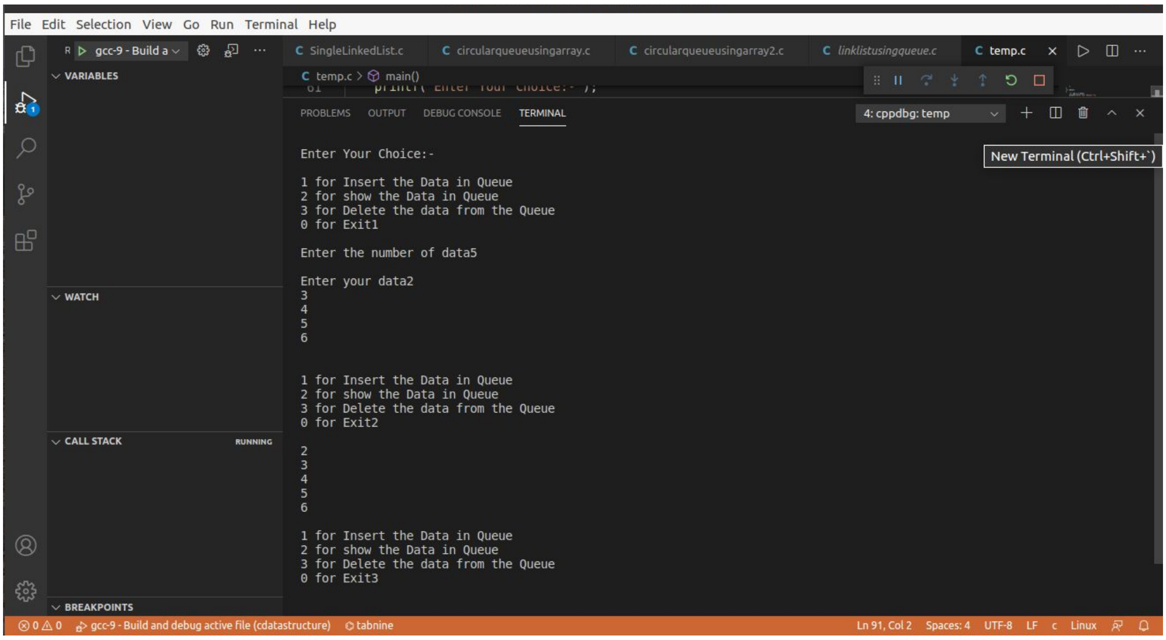
```
 1 - Enque
 2 - Deque
 3 - Front element
 4 - Empty
 5 - Exit
 6 - Display
 7 - Queue size
 Enter choice : 1
Enter data : 34

 Enter choice : 2

 Dequed value : 34
 Enter choice : 3

 No front element in Queue as queue is empty
 Enter choice : 7

 Queue size : 0
 Enter choice : 1
Enter data : 34

 Enter choice : 1
Enter data : 56

 Enter choice : 1
Enter data : 78

 Enter choice : 7

 Queue size : 3
 Enter choice : 
```

Aim: WAP to implement circular queue using linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node* next;
};
struct node *f = NULL;
struct node *r = NULL;
void enqueue(int d) //Insert elements in Queue
{
   struct node* n;
   n = (struct node*)malloc(sizeof(struct node));
   n->data = d;
   n->next = NULL;
   if((r==NULL)&&(f==NULL))
   {
       f = r = n;
       r->next = f;
   }
   else
   {
       r->next = n;
       r = n;
       n->next = f;
   }
}
void dequeue() // Delete an element from Queue
{
   struct node* t;
   t = f;
   if((f==NULL)&&(r==NULL))
       printf("\nQueue is Empty");
   else if(f == r){
       f = r = NULL;
       free(t);
   }
   else{
       f = f->next;
       r->next = f;
       free(t);
   }
```

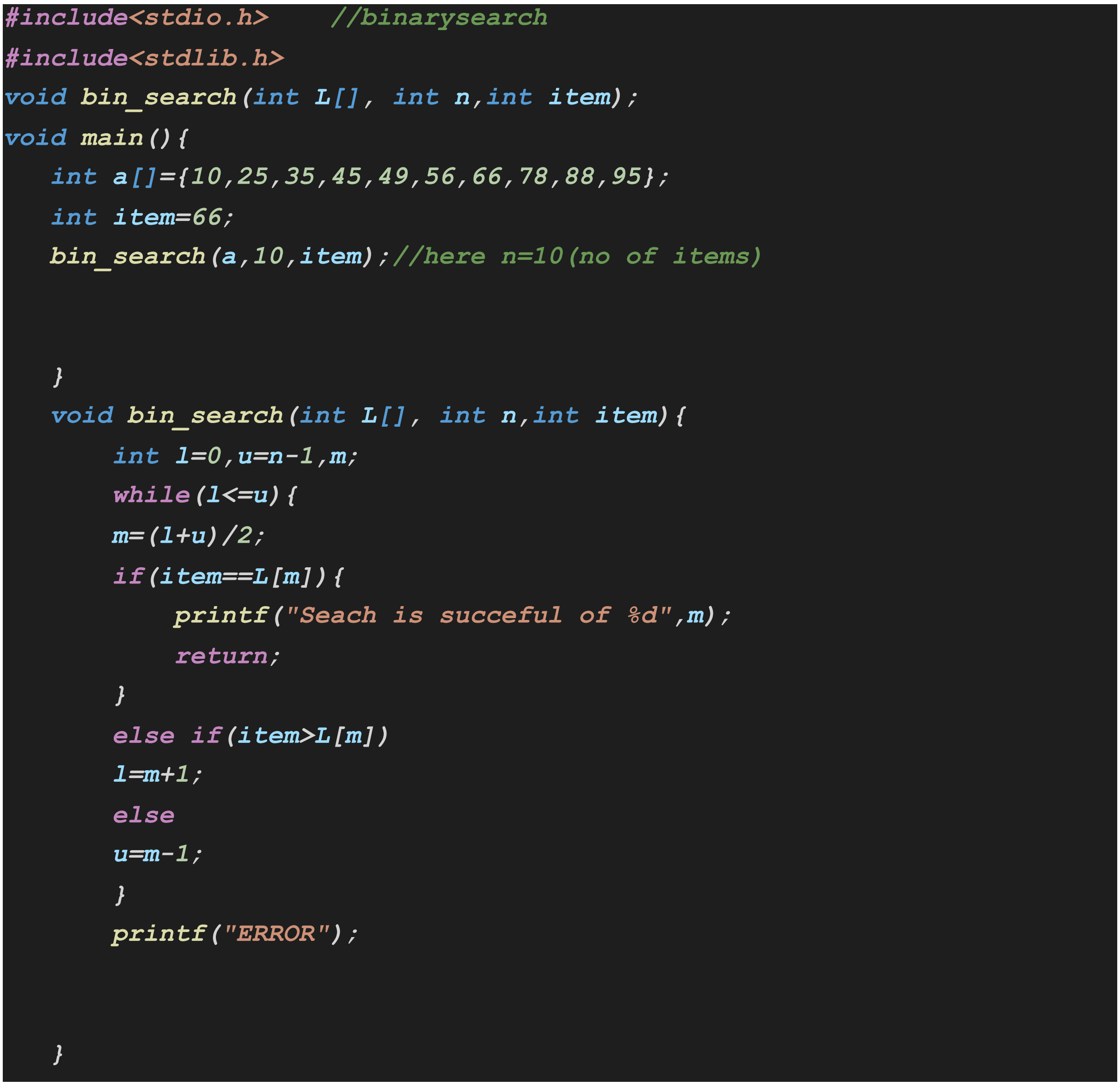
```c
}
void print(){ // Print the elements of Queue
    struct node* t;
    t = f;
    if((f==NULL)&&(r==NULL))
        printf("\nQueue is Empty");
    else{
        do{
            printf("\n%d",t->data);
            t = t->next;
        }while(t != f);
    }
}
int main()
{
    int opt,n,i,data;
    printf("Enter Your Choice:-");
    do{
        printf("\n\n1 for Insert the Data in Queue\n2 for show the Data in Queue \n3 for Delete the data from the Queue\n0 for Exit");
        scanf("%d",&opt);
        switch(opt){
            case 1:
                printf("\nEnter the number of data");
                scanf("%d",&n);
                printf("\nEnter your data");
                i=0;
                while(i<n){
                    scanf("%d",&data);
                    enqueue(data);
                    i++;
                }
                break;
            case 2:
                print();
                break;
            case 3:
                 dequeue();
                break;
            case 0:
                break;
```

```
           default:
               printf("\nIncorrect Choice");


       }
   }while(opt!=0);
return 0;
}
```

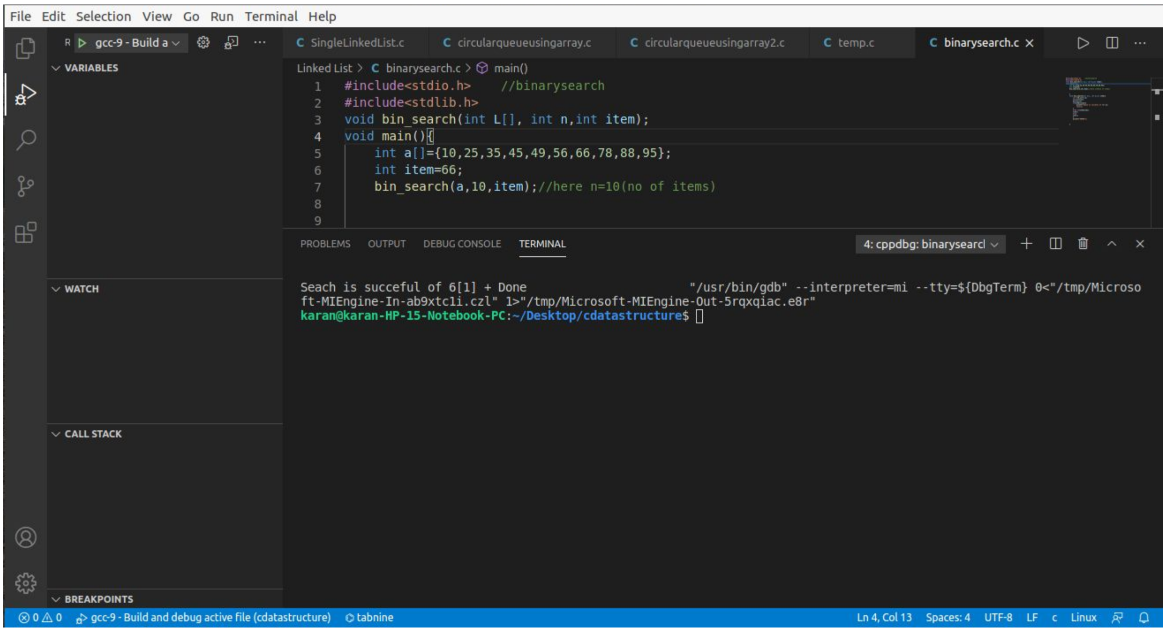


```
Enter Your Choice:-

1 for Insert the Data in Queue
2 for show the Data in Queue
3 for Delete the data from the Queue
0 for Exit1

Enter the number of data5

Enter your data2
3
4
5
6


1 for Insert the Data in Queue
2 for show the Data in Queue
3 for Delete the data from the Queue
0 for Exit2

2
3
4
5
6

1 for Insert the Data in Queue
2 for show the Data in Queue
3 for Delete the data from the Queue
0 for Exit3
```

Aim : WAP to implement binary and linear search.



```
#include<stdio.h>    //binarysearch
#include<stdlib.h>
void bin_search(int L[], int n,int item);
void main(){
   int a[]={10,25,35,45,49,56,66,78,88,95};
   int item=66;
   bin_search(a,10,item);//here n=10(no of items)


   }
   void bin_search(int L[], int n,int item){
       int l=0,u=n-1,m;
       while(l<=u){
       m=(l+u)/2;
       if(item==L[m]){
           printf("Seach is succeful of %d",m);
           return;
       }
       else if(item>L[m])
       l=m+1;
       else
       u=m-1;
       }
       printf("ERROR");


   }
```

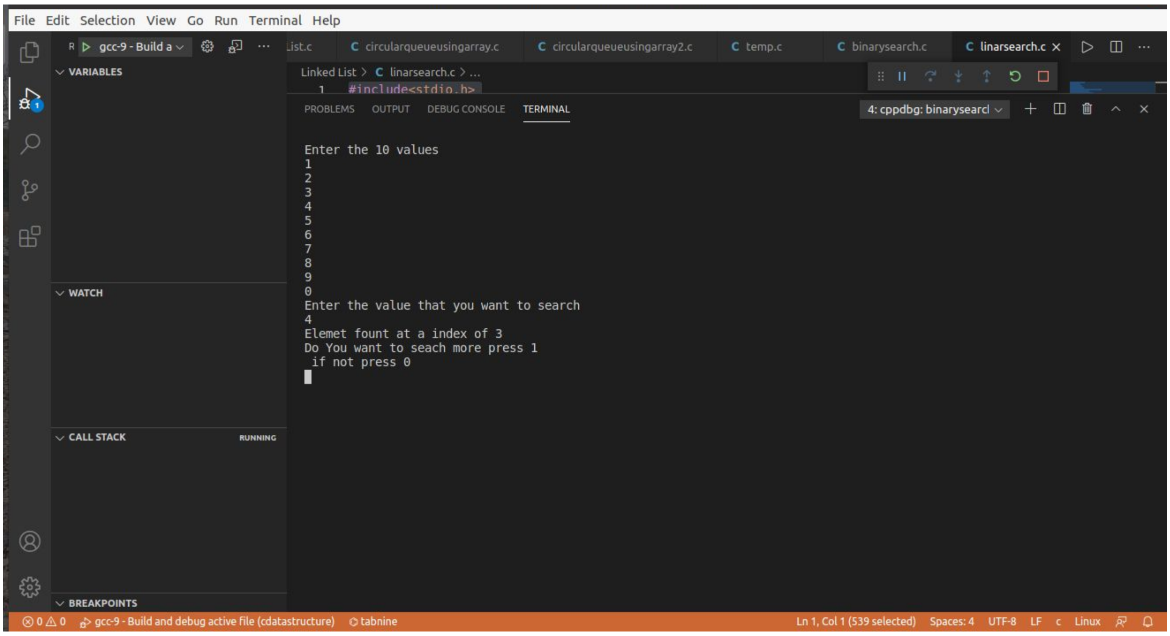


```c
#include<stdio.h>     //binarysearch
#include<stdlib.h>
void bin_search(int L[], int n,int item);
void main(){
    int a[]={10,25,35,45,49,56,66,78,88,95};
    int item=66;
    bin_search(a,10,item);//here n=10(no of items)


```

```
Seach is succeful of 6[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microso
ft-MIEngine-In-ab9xtc1i.czl" 1>"/tmp/Microsoft-MIEngine-Out-5rqxqiac.e8r"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

---> Linear seach

```
#include<stdio.h>
void main(){
   int A[10];
   int i,n,cho;
   printf("Enter the 10 values\n");
   for(i=0;i<10;i++){
       scanf("%d",&A[i]);

   }
   while(cho){
   printf("Enter the value that you want to search\n");
   scanf("%d",&n);
   for(i=0;i<10;i++){
       if(A[i]==n){
           printf("Elemet fount at a index of %d\n",i);
           break;
       }
   }
   if(i==10){
       printf("element not found\n");
   }
   printf("Do You want to seach more press 1\n if not press 0\n");
   scanf("%d",&cho);
}
}
```



File Edit Selection View Go Run Terminal Help
gcc-9 - Build a
List.c
circularqueueusingarray.c
circularqueueusingarray2.c
temp.c
binarysearch.c
linarsearch.c
Linked List > linarsearch.c > ...
VARIABLES
WATCH
CALL STACK
RUNNING
BREAKPOINTS
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
4: cppdbg: binarysearcl
gcc-9 - Build and debug active file (cdatastructure)
tabnine
Ln 1, Col 1 (539 selected) Spaces: 4 UTF-8 LF c Linux

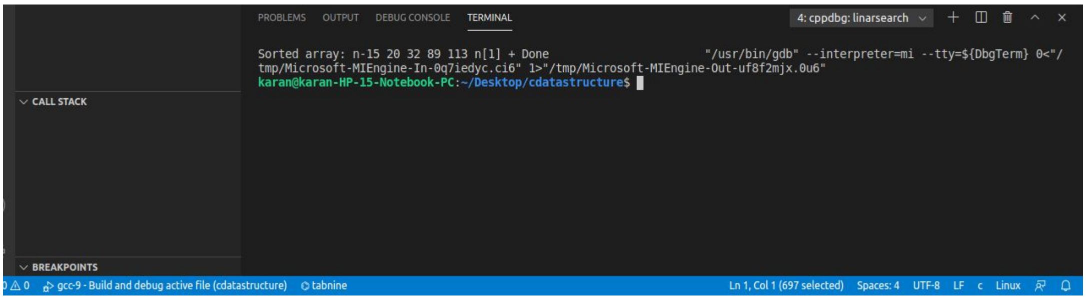
```
#include<stdio.h>
```

```
Enter the 10 values
1
2
3
4
5
6
7
8
9
0
Enter the value that you want to search
4
Elemet fount at a index of 3
Do You want to seach more press 1
 if not press 0
```

Aim: WAP to implement sorting algorithms like bubble sort,merge sort,quick sort and insertion sort.

1.Bubble sort

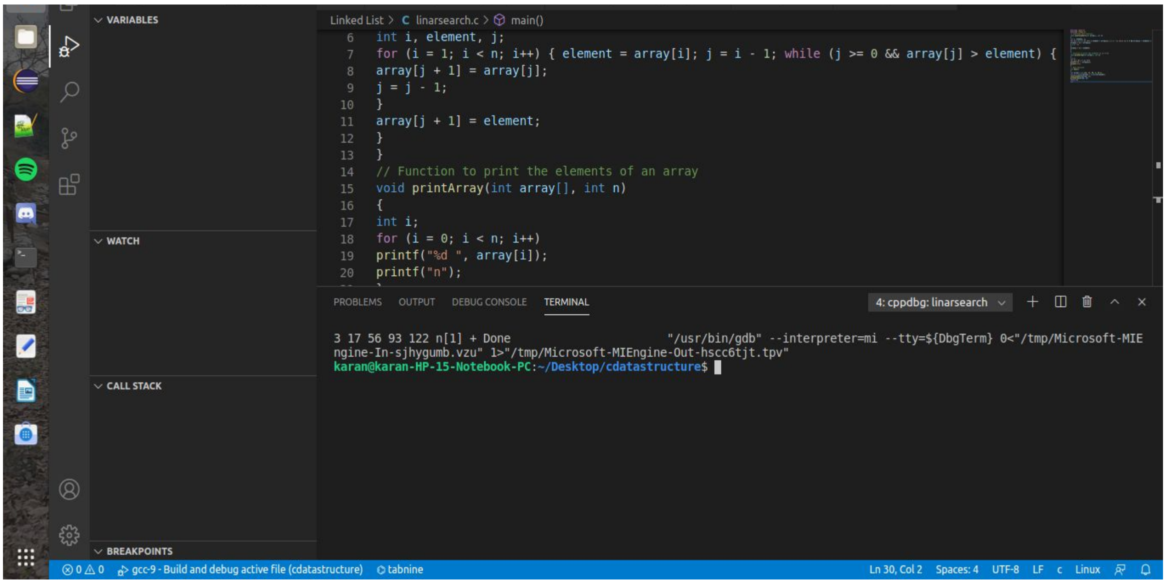
```
#include <stdio.h>
// Function to swap elements
void swap(int *a, int *b)
{
int temp = *a;
*a = *b;
*b = temp;
}
// bubble sort function
void bubbleSort(int array[], int n)
{
int i, j;
for (i = 0; i < n-1; i++)
for (j = 0; j < n-i-1; j++) if (array[j] > array[j+1])
swap(&array[j], &array[j+1]);
}
// Function to print the elements of an array
void printArray(int array[], int size)
{
int i;
for (i=0; i < size; i++)
printf("%d ", array[i]);
printf("n");
}
// Main Function
int main()
{
int array[] = {89, 32, 20, 113, -15};
int size = sizeof(array)/sizeof(array[0]);
bubbleSort(array, size);
printf("Sorted array: n");
printArray(array, size);
return 0;
}
```



```
Sorted array: n-15 20 32 89 113 n[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/
tmp/Microsoft-MIEngine-In-0q7iedyc.ci6" 1>"/tmp/Microsoft-MIEngine-Out-uf8f2mjx.0u6"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

2.*Insertion Sort*

```
#include <math.h>
#include <stdio.h>
// Insertion Sort Function
void insertionSort(int array[], int n)
{
int i, element, j;
for (i = 1; i < n; i++) { element = array[i]; j = i - 1; while (j >= 0
&& array[j] > element) {
array[j + 1] = array[j];
j = j - 1;
}
array[j + 1] = element;
}
}
// Function to print the elements of an array
void printArray(int array[], int n)
{
int i;
for (i = 0; i < n; i++)
printf("%d ", array[i]);
printf("n");
}
// Main Function
int main()
{
int array[] = { 122, 17, 93, 3, 56 };
int n = sizeof(array) / sizeof(array[0]);
insertionSort(array, n);
printArray(array, n);
return 0;
}
```



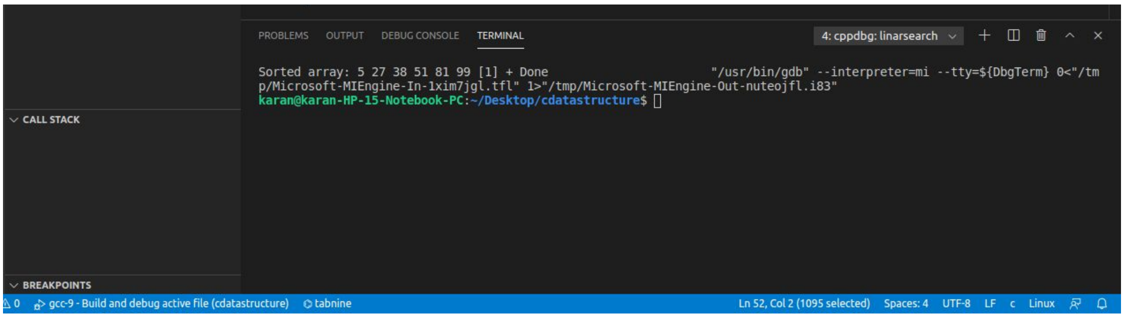
```c
int i, element, j;
for (i = 1; i < n; i++) { element = array[i]; j = i - 1; while (j >= 0 && array[j] > element) {
array[j + 1] = array[j];
j = j - 1;
}
array[j + 1] = element;
}
}
// Function to print the elements of an array
void printArray(int array[], int n)
{
int i;
for (i = 0; i < n; i++)
printf("%d ", array[i]);
printf("n");
```

```
3 17 56 93 122 n[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIE
ngine-In-sjhygumb.vzu" 1>"/tmp/Microsoft-MIEngine-Out-hscc6tjt.tpv"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

3.Quick sort

```
#include<stdio.h>
// Function to swap two elements
void swapElements(int* x, int* y)
{
int temp = *x;
*x = *y;
*y = temp;
}
// Partition function
int partition (int arr[], int lowIndex, int highIndex)
{
int pivotElement = arr[highIndex];
int i = (lowIndex - 1);
for (int j = lowIndex; j <= highIndex- 1; j++)
{
if (arr[j] <= pivotElement)
{
i++;
swapElements(&arr[i], &arr[j]);
}
}
swapElements(&arr[i + 1], &arr[highIndex]);
return (i + 1);
}
// QuickSort Function
void quickSort(int arr[], int lowIndex, int highIndex)
{
if (lowIndex < highIndex)
{
int pivot = partition(arr, lowIndex, highIndex);
// Separately sort elements before & after partition
quickSort(arr, lowIndex, pivot - 1);
quickSort(arr, pivot + 1, highIndex);
}
}
// Function to print array
void printArray(int arr[], int size)
{
int i;
for (i=0; i < size; i++)
printf("%d ", arr[i]);
}
```

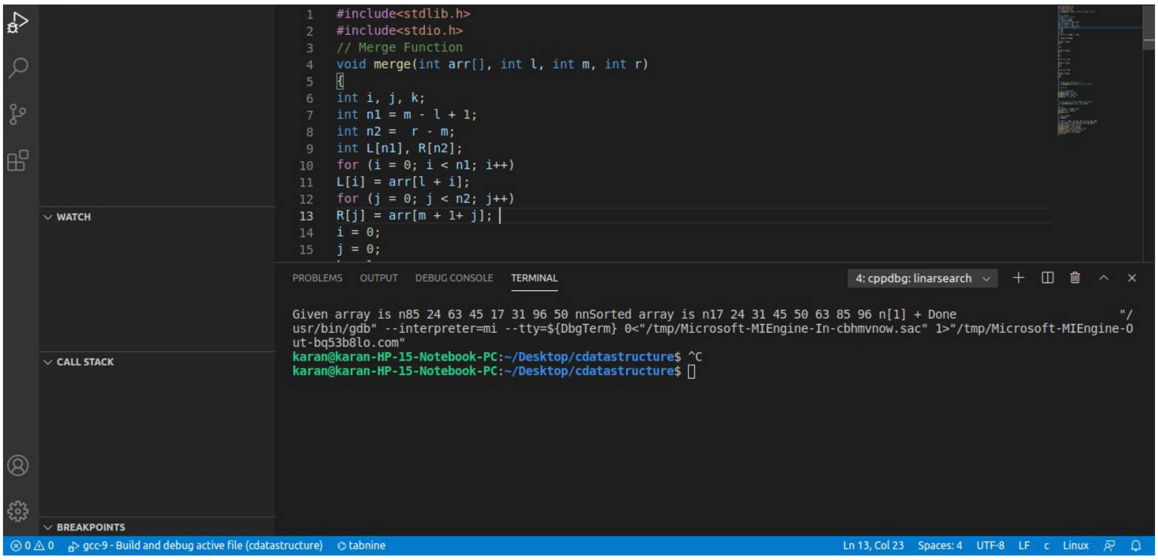
```
// Main Function
int main()
{
int arr[] = {81, 27, 38, 99, 51, 5};
int n = sizeof(arr)/sizeof(arr[0]);
quickSort(arr, 0, n-1);
printf("Sorted array: ");
printArray(arr, n);
return 0;
}
```



```
Sorted array: 5 27 38 51 81 99 [1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tm
p/Microsoft-MIEngine-In-1xim7jgl.tfl" 1>"/tmp/Microsoft-MIEngine-Out-nuteojfl.i83"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

4.Merge sort

```c
#include<stdlib.h>
#include<stdio.h>
// Merge Function
void merge(int arr[], int l, int m, int r)
{
int i, j, k;
int n1 = m - l + 1;
int n2 =  r - m;
int L[n1], R[n2];
for (i = 0; i < n1; i++)
L[i] = arr[l + i];
for (j = 0; j < n2; j++)
R[j] = arr[m + 1+ j];
i = 0;
j = 0;
k = l;
while (i < n1 && j < n2)
{
if (L[i] <= R[j])
{
arr[k] = L[i];
i++;
}
else
{
arr[k] = R[j];
j++;
}
k++;
}
while (i < n1)
{
arr[k] = L[i];
i++;
k++;
}
while (j < n2)
{
arr[k] = R[j];
j++;
k++;
}
}
// Merge Sort Function in C
void mergeSort(int arr[], int l, int r)
{
if (l < r)
{
int m = l+(r-l)/2;
mergeSort(arr, l, m);
mergeSort(arr, m+1, r);
merge(arr, l, m, r);
}
}
// Functions to Print Elements of Array
void printArray(int A[], int size)
{
int i;
for (i=0; i < size; i++)
printf("%d ", A[i]);
printf("n");
}
// Main Method
int main()
{
int arr[] = {85, 24, 63, 45, 17, 31, 96, 50};
int arr_size = sizeof(arr)/sizeof(arr[0]);
printf("Given array is n");
printArray(arr, arr_size);
mergeSort(arr, 0, arr_size - 1);
printf("nSorted array is n");
printArray(arr, arr_size);
return 0;
}
```

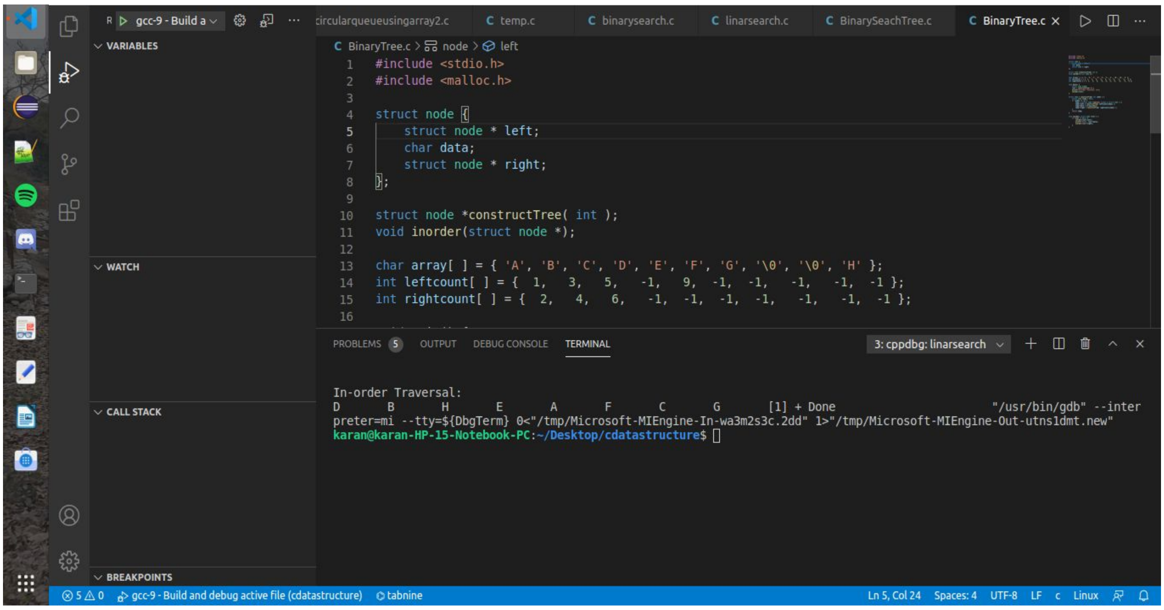


```c
#include<stdlib.h>
#include<stdio.h>
// Merge Function
void merge(int arr[], int l, int m, int r)
{
int i, j, k;
int n1 = m - l + 1;
int n2 =  r - m;
int L[n1], R[n2];
for (i = 0; i < n1; i++)
L[i] = arr[l + i];
for (j = 0; j < n2; j++)
R[j] = arr[m + 1+ j];
i = 0;
j = 0;
```

```
Given array is n85 24 63 45 17 31 96 50 nnSorted array is n17 24 31 45 50 63 85 96 n[1] + Done                 "/
usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-cbhmvnow.sac" 1>"/tmp/Microsoft-MIEngine-O
ut-bq53b8lo.com"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ ^C
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

Aim:WAP to implement a binary search tree using linked list

```
  #include <stdio.h>
#include <malloc.h>
struct node {
   struct node * left;
   char data;
   struct node * right;
};
struct node *constructTree( int );
void inorder(struct node *);
char array[ ] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', '\0', '\0', 'H' };
int leftcount[ ] = {  1,    3,    5,    -1,    9,   -1,   -1,    -1,    -1,   -1
};
int rightcount[ ] = {  2,    4,    6,    -1,   -1,   -1,   -1,    -1,    -1,
-1 };
void main() {
   struct node *root;
   root = constructTree( 0 );
   printf("In-order Traversal: \n");
   inorder(root);
}
struct node * constructTree( int index ) {
   struct node *temp = NULL;
   if (index != -1) {
       temp = (struct node *)malloc( sizeof ( struct node ) );
       temp->left = constructTree( leftcount[index] );
       temp->data = array[index];
       temp->right = constructTree( rightcount[index] );
   }
   return temp;
}
void inorder( struct node *root ) {
   if (root != NULL) {
       inorder(root->left);
       printf("%c\t", root->data);
       inorder(root->right);
   }
}
```

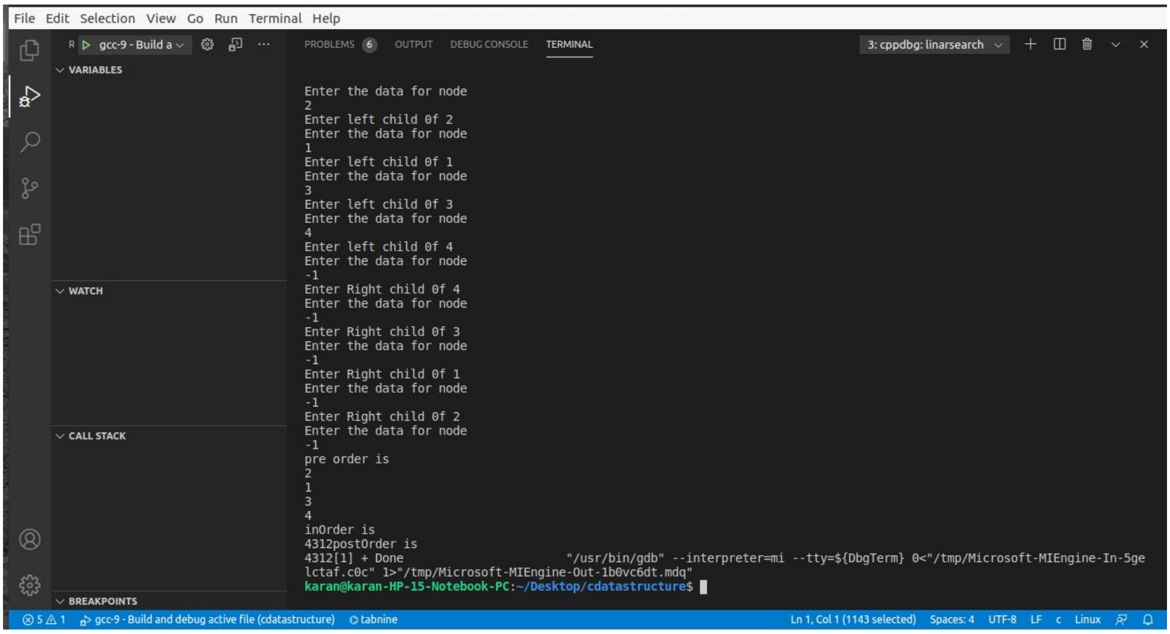


```c
#include <stdio.h>
#include <malloc.h>

struct node {
    struct node * left;
    char data;
    struct node * right;
};

struct node *constructTree( int );
void inorder(struct node *);

char array[ ] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', '\0', '\0', 'H' };
int leftcount[ ] = {  1,   3,   5,   -1,   9,  -1,  -1,   -1,   -1,  -1 };
int rightcount[ ] = {  2,   4,   6,   -1,  -1,  -1,  -1,   -1,   -1,  -1 };

```

```
In-order Traversal:
D       B       H       E       A       F       C       G       [1] + Done                       "/usr/bin/gdb" --inter
preter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-wa3m2s3c.2dd" 1>"/tmp/Microsoft-MIEngine-Out-utns1dmt.new"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

Aim:WAP to implement tree transversal using linked list.

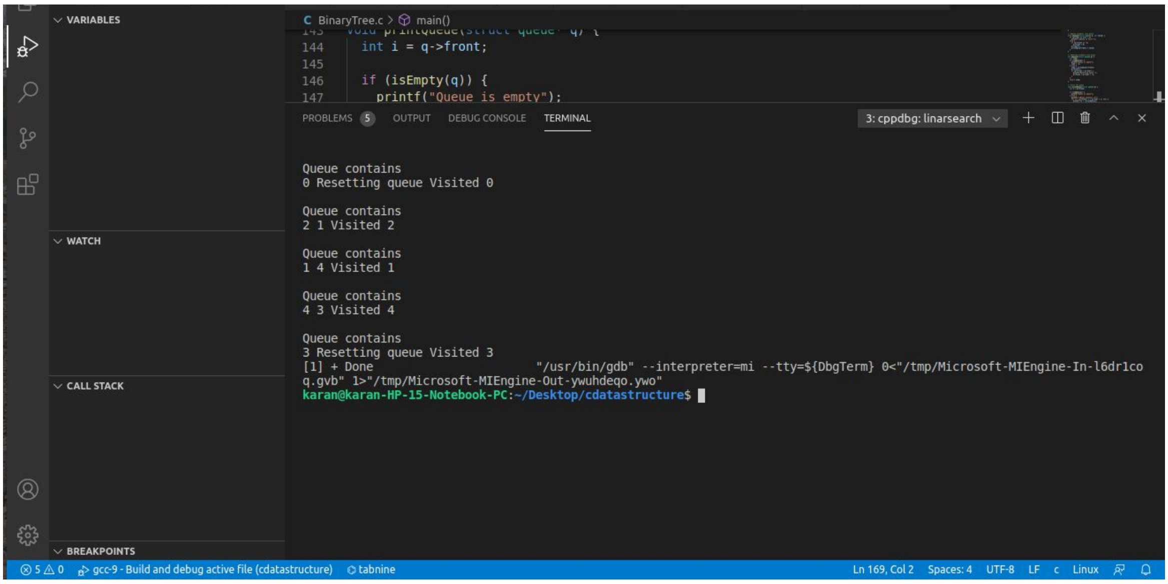
```c
#include<stdio.h>
struct node{
   int data;
   struct node *left,*right;
};
struct node* create(){
   int x;
   struct node *newnode;
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("Enter the data for node\n");
   scanf("%d",&x);
   if(x==-1){
       return(0);
   }
   newnode->data=x;
   printf("Enter left child Of %d\n",x);
   newnode->left=create();
   printf("Enter Right child Of %d\n",x);
   newnode->right=create();
   return(newnode);

}
void PreOrder(struct node *root){
   if(root==0){
       return;
   }
   printf("%d\n",root->data);
   PreOrder(root->left);
   PreOrder(root->right);
}
void inOrder(struct node *root){
   if(root==0){
       return;
   }
   inOrder(root->left);
   printf("%d",root->data);
   inOrder(root->right);

}
void postOrder(struct node *root){
     if(root==0){
       return;
   }
   postOrder(root->left);
   postOrder(root->right);
   printf("%d",root->data);
}


void main(){
struct node *root;
root=0;
root=create();
printf("pre order is\n");
PreOrder(root);
printf("inOrder is\n");
inOrder(root);
printf("postOrder is\n");
postOrder(root);
}
```



```
Enter the data for node
2
Enter left child 0f 2
Enter the data for node
1
Enter left child 0f 1
Enter the data for node
3
Enter left child 0f 3
Enter the data for node
4
Enter left child 0f 4
Enter the data for node
-1
Enter Right child 0f 4
Enter the data for node
-1
Enter Right child 0f 3
Enter the data for node
-1
Enter Right child 0f 1
Enter the data for node
-1
Enter Right child 0f 2
Enter the data for node
-1
pre order is
2
1
3
4
inOrder is
4312postOrder is
4312[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-5ge
lctaf.c0c" 1>"/tmp/Microsoft-MIEngine-Out-1b0vc6dt.mdq"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$
```

Aim: WAP to implement BFS and DFS USING LINKED LIST

```c
// BFS algorithm in C

#include <stdio.h>
#include <stdlib.h>
#define SIZE 40

struct queue {
 int items[SIZE];
 int front;
 int rear;
};

struct queue* createQueue();
void enqueue(struct queue* q, int);
int dequeue(struct queue* q);
void display(struct queue* q);
int isEmpty(struct queue* q);
void printQueue(struct queue* q);

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

struct node* createNode(int);

struct Graph {
 int numVertices;
 struct node** adjLists;
 int* visited;
};

// BFS algorithm
void bfs(struct Graph* graph, int startVertex) {
 struct queue* q = createQueue();

 graph->visited[startVertex] = 1;
 enqueue(q, startVertex);

 while (!isEmpty(q)) {
   printQueue(q);
   int currentVertex = dequeue(q);
   printf("Visited %d\n", currentVertex);

   struct node* temp = graph->adjLists[currentVertex];

   while (temp) {
     int adjVertex = temp->vertex;

     if (graph->visited[adjVertex] == 0) {
       graph->visited[adjVertex] = 1;
       enqueue(q, adjVertex);
     }
     temp = temp->next;
   }
 }
}

// Creating a node
struct node* createNode(int v) {
 struct node* newNode = malloc(sizeof(struct node));
 newNode->vertex = v;
 newNode->next = NULL;
 return newNode;
}

// Creating a graph
struct Graph* createGraph(int vertices) {
 struct Graph* graph = malloc(sizeof(struct Graph));
 graph->numVertices = vertices;

 graph->adjLists = malloc(vertices * sizeof(struct node*));
 graph->visited = malloc(vertices * sizeof(int));

 int i;
 for (i = 0; i < vertices; i++) {
   graph->adjLists[i] = NULL;
   graph->visited[i] = 0;
 }

 return graph;
}

// Add edge
void addEdge(struct Graph* graph, int src, int dest) {
 // Add edge from src to dest
 struct node* newNode = createNode(dest);
 newNode->next = graph->adjLists[src];
 graph->adjLists[src] = newNode;

 // Add edge from dest to src
 newNode = createNode(src);
 newNode->next = graph->adjLists[dest];
 graph->adjLists[dest] = newNode;
}

// Create a queue
struct queue* createQueue() {
 struct queue* q = malloc(sizeof(struct queue));
 q->front = -1;
 q->rear = -1;
 return q;
}

// Check if the queue is empty
int isEmpty(struct queue* q) {
 if (q->rear == -1)
   return 1;
 else
   return 0;
}

// Adding elements into queue
void enqueue(struct queue* q, int value) {
 if (q->rear == SIZE - 1)
   printf("\nQueue is Full!!");
 else {
   if (q->front == -1)
     q->front = 0;
   q->rear++;
   q->items[q->rear] = value;
 }
}

// Removing elements from queue
int dequeue(struct queue* q) {
 int item;
 if (isEmpty(q)) {
   printf("Queue is empty");
   item = -1;
 } else {
   item = q->items[q->front];
   q->front++;
   if (q->front > q->rear) {
     printf("Resetting queue ");
     q->front = q->rear = -1;
   }
 }
 return item;
}

// Print the queue
void printQueue(struct queue* q) {
 int i = q->front;

 if (isEmpty(q)) {
   printf("Queue is empty");
 } else {
   printf("\nQueue contains \n");
   for (i = q->front; i < q->rear + 1; i++) {
     printf("%d ", q->items[i]);
   }
 }
}

int main() {
 struct Graph* graph = createGraph(6);
 addEdge(graph, 0, 1);
 addEdge(graph, 0, 2);
 addEdge(graph, 1, 2);
 addEdge(graph, 1, 4);
 addEdge(graph, 1, 3);
 addEdge(graph, 2, 4);
 addEdge(graph, 3, 4);

 bfs(graph, 0);

 return 0;
}
```



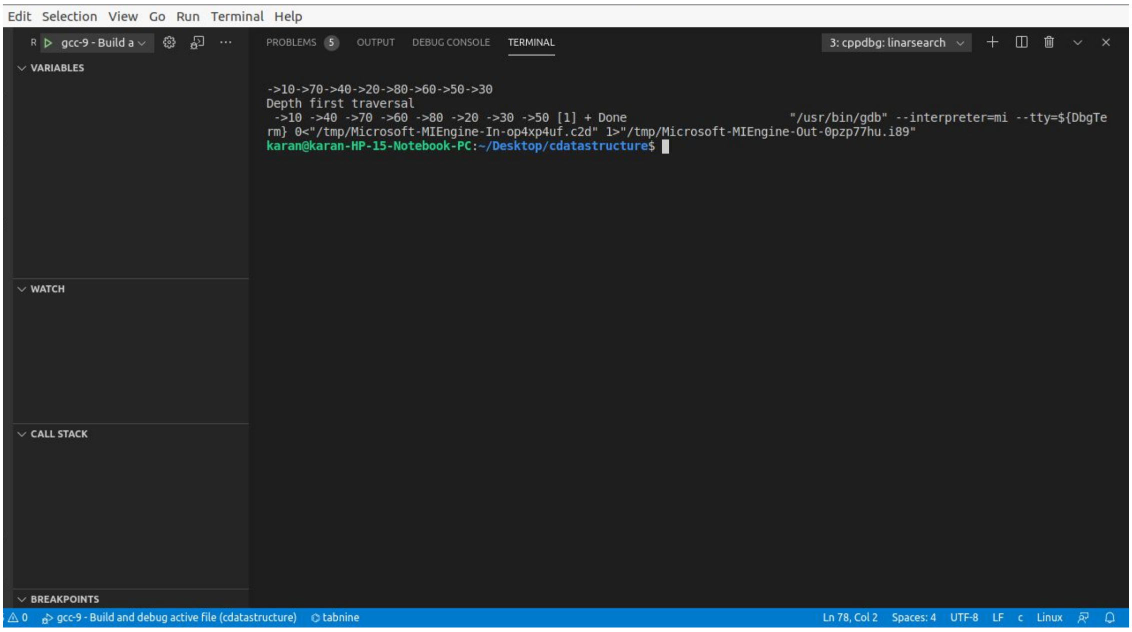
```c
void printQueue(struct queue* q) {
  int i = q->front;

  if (isEmpty(q)) {
    printf("Queue is empty");
```

```
Queue contains
0 Resetting queue Visited 0

Queue contains
2 1 Visited 2

Queue contains
1 4 Visited 1

Queue contains
4 3 Visited 4

Queue contains
3 Resetting queue Visited 3
[1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-l6dr1co
q.gvb" 1>"/tmp/Microsoft-MIEngine-Out-ywuhdeqo.ywo"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```

2.DFS

```
#include <stdio.h>
#include <stdlib.h>
struct btnode {
   int value;
   struct btnode *l;
   struct btnode *r;
};
typedef struct btnode bt;
bt *root;
bt *new, *list;
bt *create_node();
void display(bt *);
void construct_tree();
void dfs(bt *);
void main()
{
   construct_tree();
   display(root);
   printf("\n");
   printf("Depth first traversal\n ");
   dfs(root);
}
/* Creates an empty node */
bt * create_node()
{
   new=(bt *)malloc(sizeof(bt));
   new->l = NULL;
   new->r = NULL;
}
/* Constructs a tree */
void construct_tree()
{
   root = create_node();
   root->value = 50;
   root->l = create_node();
   root->l->value = 20;
   root->r = create_node();
   root->r->value = 30;
   root->l->l = create_node();
   root->l->l->value = 70;
   root->l->r = create_node();
   root->l->r->value = 80;
```

```
    root->l->r->r = create_node();
    root->l->r->r->value = 60;
    root->l->l->l = create_node();
    root->l->l->l->value = 10;
    root->l->l->r = create_node();
    root->l->l->r->value = 40;
}
/* Display the elements in a tree using inorder */
void display(bt * list)
{
    if (list == NULL)
    {
        return;
    }
    display(list->l);
    printf("->%d", list->value);
    display(list->r);
}
/* Dfs traversal using post order */
void dfs(bt * list)
{
    if (list == NULL)
    {
        return;
    }
    dfs(list->l);
    dfs(list->r);
    printf("->%d ", list->value);
}
```



```
->10->70->40->20->80->60->50->30
Depth first traversal
 ->10 ->40 ->70 ->60 ->80 ->20 ->30 ->50 [1] + Done                       "/usr/bin/gdb" --interpreter=mi --tty=${DbgTe
rm} 0<"/tmp/Microsoft-MIEngine-In-op4xp4uf.c2d" 1>"/tmp/Microsoft-MIEngine-Out-0pzp77hu.i89"
karan@karan-HP-15-Notebook-PC:~/Desktop/cdatastructure$ 
```