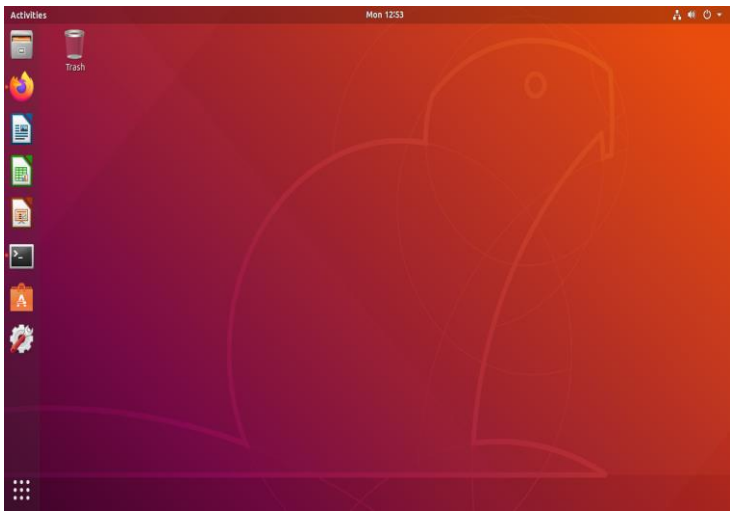
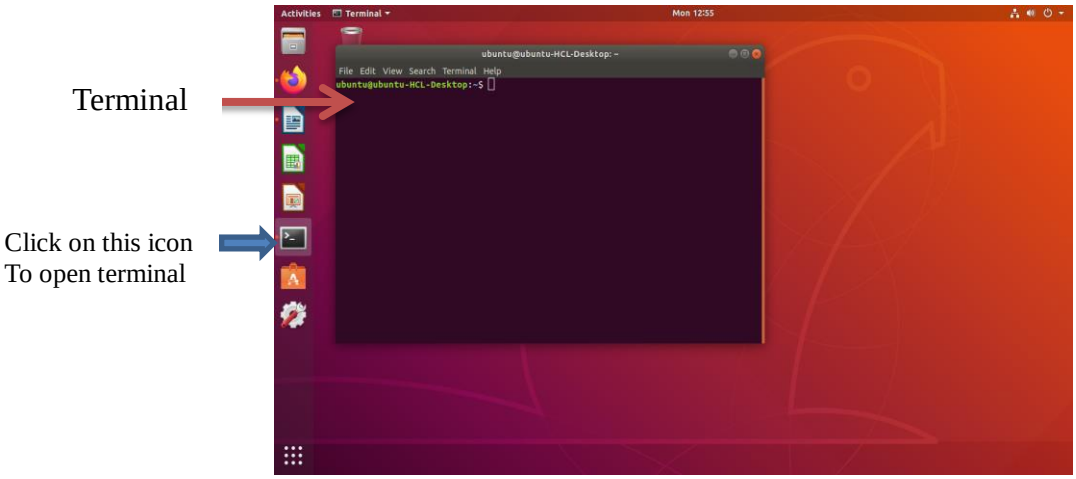
HOW TO WRITE AND RUN C PROGRAM IN UBUNTU/ LINUX

Step 1: Login to Ubuntu

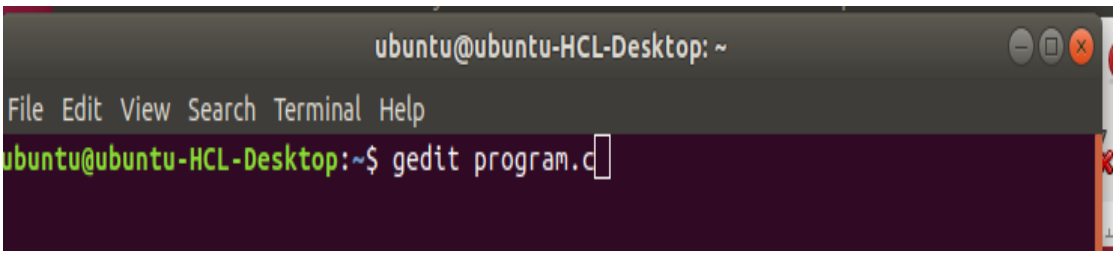


Ubuntu Desktop

Step 2: Open a terminal window, for this go to **Dash home/Side Panel> Terminal**, as shown in below image.



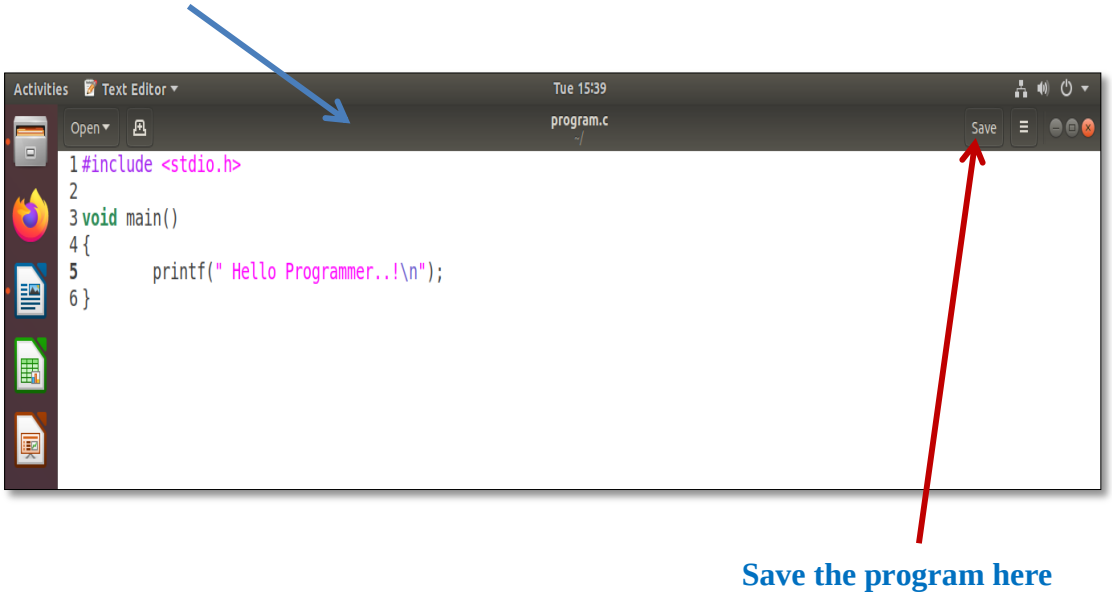
Step 3: Open Text Editor, type and save the program



Type the command to open text editor: **`gedit filename.c`**

`Note: filename: name given to C file`

Text Editor: gedit

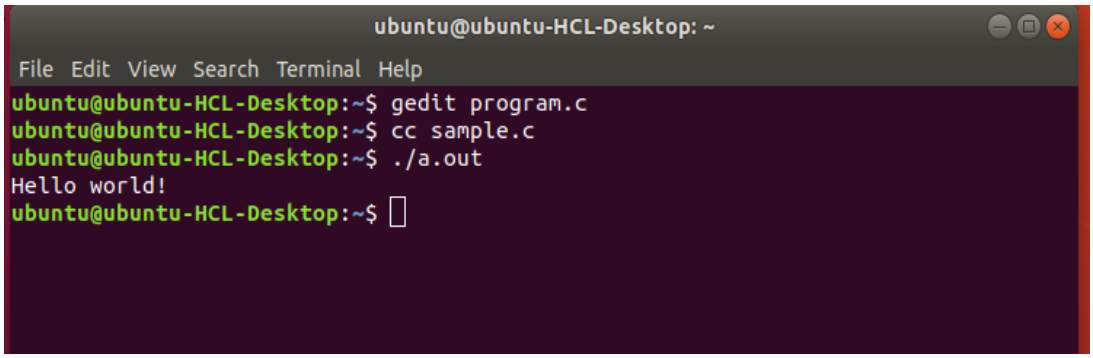


Save the program here

- **Type the code in code area**
- **Save and close the text editor**

Code Area: Type code here

Step 4: Compile and Execute the Program



- To compile C Program type the following command in terminal
  cc filename.c
- To Execute C Program type the following command in terminal
  ./a.out

Course Title	DATA STRUCTURES LABORATORY		
Course Code	22CS305	L-T-P-C	(0-0-1)1
Exam Hrs.	3	Hours / Week	2
SEE	50 Marks	Total Hours	28

Course Objective: Design and implement various data structures.

Course Outcomes (COs): Upon the completion of the course the students will be able to:

#	Course Outcomes	Mapping to POs	Mapping to PSOs
1.	Develop, design and document C programs to implement structures and pointers.	1,2,3	-
2.	Illustrate the concepts of data structures: stack, queue, linked list and trees in C using static or dynamic memory allocation and documentthem.	1,2,3	2
3.	Demonstrate the concept of recursion by developing recursive C programs and document them.	1,2,3	2

Course Contents:

	Practice Programs
1.	Write a C program to find the maximum and minimum element in an array of n integers. Use only pointers for referencing the array.
2.	Write a C program for Dynamic Memory allocation of 10 elements and find the largest element.
3.	Write a C program to represent a complex number using structure variable. Write user defined functions that accept two complex numbers and finds their sum and difference.
4.	Define a structure Author name with fields: First name, Middle name and Last name. Using the above structure, design another structure Book: ISBN, Author name, Book Title, Price, Publisher, and Edition. Write a function to search a book given the Author name. Using the above function write a C Program to store N books information and display the details of a book given the author name.
	Exercise Programs
1.	Files are placed one over another in my study room. The file which is at the top is the first one to be removed, i.e. the file which has been placed at the bottom most position remains in the pile of files for the longest period of time. Help me out to add a file and remove the bottom most file from the pile of files.
2.	Consider an algebraic expression which needs to be evaluated by a computer system. Operating System (OS) consumes less time to evaluate if it is in postfix form of the expression. Thus, help your OS to evaluate by converting the expression into its postfix form.
3.	Assume you have converted an algebraic expression into its postfix form to process the expression fast.This expression need to be evaluated for a given set of values. Implement the above.
4.	Assume you come across a toll gate while you are on your way to home town. Illustrate the workingof the toll gate using suitable data structure
5.	1. Suppose you want to search a text book in a huge library where books are arranged in alphabetical order. Optimize your search by using recursion. 2. Implement Tower of Hanoi problem using recursion.
6	Consider a traffic signal controlled by a computer system. Traffic signal has three colors: Red, yellow and Green. All these glows in a circular fashion based on the traffic. Implement the above using suitable data structure.

7	The parking lot has a fixed number of parking spaces. Cars can enter the parking lot and occupy an available space, and they can also exit the parking lot, freeing up the space for other cars. Designing a parking lot management system using a circular queue.
8	Consider a treasure hunt task where a series of clues are given. Clue1 gives hint to clue2, clue2 Provides hint for clue3 and so on until you can get a hint to the final treasure. Develop an illustrationto demonstrate the above scenario.
9	Consider a list of numbers. Find ➢ Maximum number ➢ Minimum number ➢ Sum of all the numbers
10	The phonebook will contain a list of contacts sorted in ascending order based on their names. Each contact will have a name and a phone number. Developing a phonebook management system using an ordered linked list.
11	Assume you have an iPod, where in you have stored plenty of songs so that you get engaged during a long journey. If you want to hear a particular song, you need to use forward button to reach that song and can also traverse back using backward button. Implement the following using relevant data structure.
12	Your text book contains chapters, sections, subsections, subdivisions, etc. Illustrate this scenario of text book using tree structure.

Experiment 1:

Files are placed one over another in my study room. The file which is at the top is the first one to be removed, i.e. the file which has been placed at the bottom most position remains in the pile of files for the longest period of time. Help me out to add a file and remove the bottom most file from the pile of files.

```
#include <stdio.h>
#include <stdlib.h>
#define max 4  // stack size id set to 4

int s[max];
int top=-1;

void push(int s[],int val);
void pop(int s[]);
void display(int s[]);

void main()
{
     int val,option;

     do
     {
          printf("\n****Stack Operations****\n");
          printf("1. push\t2. pop\t3. display\t4. Exit\n");
          printf("enter your option:");
          scanf("%d",&option);
          switch(option)
          {
               case 1: printf("\nEnter the number to be\n");
                       scanf("%d",&val);
                       push(s,val);
                       break;
               case 2: pop(s);
                       break;
               case 3: display(s);
                       break;
               case 4: exit(0);
                       break;
          }
     }while(option!=4);
}
void push(int s[],int val)
{
     if(top==max-1)
     {
          printf("STACK IS OVERFLOW\n");
```

```
      }
      else
      {
            top=top+1;
            s[top]=val;
            printf("%d is inserted into stack\n",val);
      }
}
void pop(int s[])
{
      int val;
      if(top==-1)
      {
            printf("STACK IS UNDERFLOW\n");
      }
      else
      {
            val=s[top];
            top=top-1;
            printf("%d is deleted from stack\n",val);
      }
}
void display(int s[])
{
      int i;
      if(top==-1)
      {
            printf("STACK IS EMPTY\n");
      }
      else
      {
            printf("Contents of stack are\n");
            for(i=top;i>=0;i--)
            {
                  printf("%d\n",s[i]);
            }
      }
}
```

Output:

To Compile: **cc prog1.c**

To Execute: **./a.out**

```
****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  1
Enter the number to be inserted into stack
10
10 is inserted into stack
```

```
****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  1
Enter the number to be inserted into stack
20
20 is inserted into stack

****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  1
Enter the number to be inserted into stack
30
30 is inserted into stack

****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  1
Enter the number to be inserted into stack
40
STACK IS OVERFLOW

****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  3
Contents of stack are
30
20
10
****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  2
30 is deleted from stack

****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  2
20 is deleted from stack

****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  2
10 is deleted from stack

****Stack Operations****
1. push 2. Pop 3. Display 4. Exit
enter your option:  2
STACK IS UNDERFLOW
```

Experiment 2:

Consider an algebraic expression which needs to be evaluated by a computer system. Operating System (OS) consumes less time to evaluate if it is in postfix form of the expression. Thus, help your OS to evaluate by converting the expression into its post form.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#define max 100
char s[max];
int top=-1;

void push(char s[],char);
char pop(char s[]);
void InfixToPostfix(char infix[],char postfix[]);
int getPriority(char);

int main()
{
     char infix[100],postfix[100];

     printf("Enter the valid infix expression\n");
     scanf("%s",infix);
     strcpy(postfix,"");
     InfixToPostfix(infix,postfix);
     printf("\n The postfix expression is:\n");
     printf("%s\n",postfix);
     return 0;
}

// function to convert infix expression into postfix
expression
void InfixToPostfix(char infix[],char postfix[])
{
     int i=0,j=0;
     char temp;
     strcpy(postfix,"");
     /*Repeat until each character in infix exp.is scanned */
     while(infix[i]!='\0')
     {
          /* if '(' is encountered */
          if(infix[i]=='(')
          {
               //push current symbol on to stack
               push(s,infix[i]);
```

```
                i++;
            }
            //if ')' is encountered
            else if(infix[i]==')')
            {
               /*repeatedly pop the elements from stack and add
               that symbol into postfix exp. until a '(' is
               encountered*/
                while((top!=-1)&&(s[top]!='('))
                {
                     postfix[j]=pop(s);
                     j++;
                }

                if(top==-1)
                {
                     printf("\n Incorrect Expression\n");
                     exit(0);
                }
                //remove the '(' from stack
                temp=pop(s);
                i++;
            }
            /*if an operand(digit or charecter) is Encountered*/
            else if(isdigit(infix[i])||isalpha(infix[i]))
            {
                 //add the current symbol into postfix exp.
                postfix[j]=infix[i];
                j++;
                i++;
            }
            // if an operator is encountered
            else if(infix[i]=='+'|| infix[i]=='-'||infix[i]=='*'
||infix[i]=='/'||infix[i]=='%'||infix[i]=='$'||infix[i]=='^')
            {
            /*repeatedly pop from stack and add each operator to
             the postfix exp. if top of stack has same precedence
             or a higher precedence than input symbol(input
             operator)*/
               while((top!=-1) && (s[top]!='(') &&
               (getPriority(s[top])>=getPriority(infix[i])))
               {
                   postfix[j]=pop(s);
                   j++;
               }

               // otherwise push that operator to the stack
              push(s,infix[i]);
```

```
            i++;
        }
        else
        {
            printf("\n Incorect Element in Expression\n");
            exit(1);
        }
    }

    /*repeatedly pop from stack and add each operator to the
      postfix exp.*/

    while((top!=-1)&&(s[top]!='('))
    {
        postfix[j]=pop(s);
        j++;
    }
    postfix[j]='\0';
}
//function to get the priority of an operator
int getPriority(char op)
{
    if(op=='^'|| op=='$')
        return 2;
    else if(op=='/' ||op=='*' ||op=='%')
        return 1;
    else if(op=='+' || op=='-')
        return 0;
}

void push(char s[],char val)
{
    if(top==max-1)
    {
        printf("STACK OVERFLOW\n");
    }
    else
    {
        top=top+1;
        s[top]=val;
    }
}
char pop(char s[])
{
    char val=' ';
    if(top==-1)
    {
        printf("STACK UNDERFLOW\n");
```

```
        }
        else
        {
                val=s[top];
                top=top-1;
        }
        return val;
}
```

Output:

To Compile: **cc prog2.c**

To Execute: **./a.out**

```
Enter the valid infix expression
(a+b)*c
The postfix expression is:
ab+c*
```

Experiment 3:

Assume you have converted an algebraic expression into its postfix form to process the expression fast. This expression need to be evaluated for a given set of values. Implement the above.

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

double s[max];
int top;

void push(double s[],double val);
double pop(double s[]);
double evaluate(char postfix[]);
void main()
{
     double val;
     char postfix[100];
     printf("enter the postfix expression\n");
     scanf("%s",postfix);
     val=evaluate(postfix);
     printf("result of postfix expression is %lf\n",val);
}
double evaluate(char postfix[])
{
     int i=0;
     double op1,op2,value;
     while(postfix[i]!='\0')
     {
          if(isdigit(postfix[i]))
          {
               push(s,(double)(postfix[i]-'0'));
          }
          else
          {
               op2=pop(s);
               op1=pop(s);
               switch(postfix[i])
               {
                    case '+':value=op1+op2;
                             break;
                    case '-':value=op1-op2;
                             break;
                    case '*':value=op1*op2;
                             break;
```

```
                    case '/':value=op1/op2;
                             break;
                    case '%':value=(int)op1%(int)op2;
                             break;
                    case '$':
                    case '^':value=pow(op1,op2);
                             break;
                }
                push(s,value);
            }
            i++;
        }
        return pop(s);
}
void push(double s[],double val)
{
        if(top==max-1)
        {
            printf("STACK OVERFLOW\n");
        }
        else
        {
            top=top+1;
            s[top]=val;
        }
}
double pop(double s[])
{
        char val=' ';
        if(top==-1)
        {
            printf("STACK UNDERFLOW\n");
        }
        else
        {
            val=s[top];
            top=top-1;
        }
        return val;
}
```

Output:

To Compile: **cc prog3.c -lm**
To Execute: **./a.out**

```
enter the postfix expression
64+2*
result of postfix expression is 20.000000
```

Experiment 4:

Assume you come across a toll gate while you are on your way to home town. Illustrate the working of the toll gate using suitable data structure.

```
#include <stdio.h>
#define max 3
int q[max];
int f=0,r=-1;
void insert_rear(int q[], int val);
void delete_front(int q[]);
void display(int q[]);
void main()
{
    int option,val;
    do{
       printf("\n******Queue Operation*********\n");
       printf("1.insert\t2.delete\t3.display\t4.exit\n");
       printf("enter your option: ");
       scanf("%d",&option);
       switch(option)
       {
         case 1: printf("\nenter the number to be inserted\n");
                 scanf("%d",&val);
                 insert_rear(q,val);
                 break;
         case 2: delete_front(q);
                 break;
         case 3: display(q);
                 break;
       }
    }while(option!=4);
}
void insert_rear(int q[], int val)
{
    if(r==max-1)
       printf("queue is overflow\n");
    else
    {
       r=r+1;
       q[r]=val;
       printf("%d is inserted into queue\n",val);
    }
}
void delete_front(int q[])
{
    int val;
    if(f>r)
        printf("queue is underflow\n");
```

```
    else
    {
        val=q[f];
        f=f+1;
        printf("%d is deleted from queue\n",val);
    }
}
void display(int q[])
{
     int i;
     if(f>r)
         printf("queue is empty\n");
     else
     {
         printf("content of the queue are\n");
         for(i=f;i<=r;i++)
         {
             printf("%d\t",q[i]);
         }
     }
}
```

Output:

To Compile: **cc prog4.c**

To Execute: **./a.out**

```
******Queue Operation**********
1.insert  2.delete  3.display  4.exit
enter your option: 1
enter the number to be inserted
10
10 is inserted into queue

******Queue Operation**********
1.insert  2.delete  3.display  4.exit
enter your option: 1
enter the number to be inserted
20
20 is inserted into queue

******Queue Operation**********
1.insert  2.delete  3.display  4.exit
enter your option: 1
enter the number to be inserted
30
30 is inserted into queue

******Queue Operation**********
1.insert  2.delete  3.display  4.exit
```

```
enter your option: 1
enter the number to be inserted
40
queue is overflow

******Queue Operation**********
1.insert  2.delete  3.display  4.exit
enter your option:  2
10 is deleted from queue

******Queue Operation**********
1.insert  2.delete  3.display  4.exit
enter your option:  2
20 is deleted from queue

******Queue Operation**********
1.insert  2.delete  3.display  4.exit
enter your option:  2
30 is deleted from queue

******Queue Operation**********
1.insert  2.delete  3.display  4.exit
enter your option:  2
queue is underflow
```

Experiment 5a:

Suppose you want to search a text book in a huge library where books are arranged in alphabetical order. Optimize your search by using recursion.

```
#include <stdio.h>
int search(int key, int a[20],int low, int high)
{
   int mid;
   if(low>high)
      return -1;
   mid=(low+high)/2;
   if(key==a[mid])
      return mid;
   if(key<a[mid])
      return search(key,a,low,mid-1);
   else
      return search(key,a,mid+1,high);
}
void main()
{
   int n,i,key,pos,a[20];
   printf("enter the number of elements\n");
   scanf("%d",&n);
   printf("enter %d numbers\n",n);
   for(i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   printf("enter the element to be searched\n");
   scanf("%d",&key);
   pos=search(key,a,0,n-1);
   if(pos==-1)
      printf("key element not found\n");
   else
      printf("key element found in position : %d \n",(pos+1));
}
```

Output:

To Compile: **cc prog11a.c**

To Execute: **./a.out**

```
enter the number of elements
4
enter 4 numbers
11
22
33
44
enter the element to be searched
33
key element found in  position : 3
```

Experiment 5b:

Implement Tower of Hanoi problem using recursion.

```
#include <stdio.h>
void move(int n,char src, char dest, char temp)
{
   if(n==1)
   {
      printf("Move one disc from %c to %c\n",src,dest);
      return;
   }
   else
   {
      move(n-1,src,temp,dest);
      move(1,src,dest,temp);
      move(n-1,temp,dest,src);
   }
}
void main()
{
   int n;
   printf("enter the number of discs\n");
   scanf("%d",&n);
   move(n,'A','C','B');
}
```

To Compile: **cc prog11b.c**

To Execute: **./a.out**

```
enter the number of discs
3
Move one disc from A to C
Move one disc from A to B
Move one disc from C to B
Move one disc from A to C
Move one disc from B to A
Move one disc from B to C
Move one disc from A to C
```

Experiment 6:

Consider a traffic signal controlled by a computer system. Traffic signal has three colours: Red, Yellow and Green. All these glows in a circular fashion based on the traffic. Implement the above using suitable data structure.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5

int f=0;
int r=-1;
int q[MAX];
int count=0;

void insert(int q[],int val)
{
     if(count==MAX)
          printf("\n Queue Overflow\n");
     else
     {
          r=(r+1)%MAX;
          q[r]=val;
          count=count+1;
          printf("\n %d is inserted\n", val);
     }
}
void delete(int q[])
{
     if(count==0)
          printf("\n Queue Underflow");
     else
     {
          printf("\n %d is deleted\n", q[f]);
          f=(f+1)%MAX;
          count=count-1;
     }
}
void display(int q[])
{
     int i,k;
     if(count==0)
          printf("Queue Empty\n");
     else
     {
          k=f;
          for(i=1;i<=count;i++)
          {
               printf("%d\t",q[k]);
```

```
                k=(k+1)%MAX;
            }
        }
}

void main()
{
     int option,val;
     do
     {
          printf("\n***Circular queue operations***");
          printf("\n1.Insert\t2.Delete\t3.Display\t4.Exit\n");
          printf("Enter your choice=");
          scanf("%d",&option);
          switch(option)
          {
               case 1: printf("\nThe value to insert\n");
                       scanf("%d",&val);
                       insert(q,val);
                       break;
               case 2: delete(q);
                       break;
               case 3: printf("Queue contains\n");
                       display(q);
                       break;
               case 4: exit(0);
          }
     }while(option!=4);
}
```

Outout:

To Compile: **cc prog5.c**

To Execute: **./a.out**

```
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  1
enter the number to be inserted
11
11 is inserted into queue
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  1
enter the number to be inserted
22
22 is inserted into queue
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  1
```

```
enter the number to be inserted
33
33 is inserted into queue
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  1
enter the number to be inserted
44
44 is inserted into queue
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  1
enter the number to be inserted
55
queue is overflow
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  3
content of the queue are
11   22   33   44
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  2
11 is deleted from queue
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  2
22 is deleted from queue
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  2
33 is deleted from queue

*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  2
44 is deleted from queue
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  2
queue is underflow
*********Circular Queue Operation************
1.insert  2.delete  3.display  4.exit
enter your option:  3
queue is empty
```

Experiment 7:

The parking lot has a fixed number of parking spaces. Cars can enter the parking lot and occupy an available space, and they can also exit the parking lot, freeing up the space for other cars. Designing a parking lot management system using a circular queue.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *link;
};
typedef struct node *NODE;
NODE first=NULL;

NODE insertRear(NODE first)
{
   NODE temp,cur;
   int item;
   printf("\nThe value to insert\n");
   scanf("%d",&item);
   temp=(NODE) malloc(sizeof(NODE));
   if(temp==NULL)
   {
      printf("out of memory\n");
      exit(0);
   }

   temp->info=item;
   temp->link=temp;
   if(first==NULL)
   {
          first=temp;
          return first;
   }
   cur=first;
   while(cur->link!=first)
   {
          cur=cur->link;
   }
   cur->link=temp;
   temp->link=first;
```

```
   printf("%d is inserted into list\n",temp->info);;
   return first;
}

NODE deleteFront(NODE first)
{
     NODE cur;
     if(first==NULL)
     {
          printf("List is empty");
          return first;
     }
   if(first->link==first)
   {
      printf("%d is deleted from list\n",first->info);
      free(first);
      first=NULL;
      return first;
   }
   cur=first;
   while(cur->link!=first)
   {
          cur=cur->link;
   }
   cur->link=first->link;
   printf("%d is deleted from list\n",first->info);
   free(first);
   first=cur->link;
   return first;
}
void display(NODE first)
{
   NODE cur;
   if(first==NULL)
   {
      printf("List is empty\n");
      exit(0);
   }
   printf("Contents of the list are\n");
   cur=first;
   while(cur->link!=first)
   {
```

```
      printf("%d\t",cur->info);
      cur=cur->link;
   }
   printf("%d\t",cur->info);
   printf("\n");
}

void main()
{
   int option,item;
   do
   {
      printf("\n**********Linked List operations*********\n");
      printf("1.Insert \t2.Delete\t3.Display\t4.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&option);
      switch(option)
      {
         case 1:first=insertRear(first);
                break;
         case 2:first=deleteFront(first);
               break;
         case 3:display(first);
                break;
         case 4:exit(0);
      }
   }while(option!=6);
}
```

Outout:

```
**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 1

The value to insert
24

**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 1

The value to insert
45
45 is inserted into list
```

```
**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 1

The value to insert
32
32 is inserted into list

**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 1

The value to insert
82
82 is inserted into list

**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 3
Contents of the list are
24      45      32      82

**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 2
24 is deleted from list

**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 2
45 is deleted from list

**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 2
32 is deleted from list

**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 2
82 is deleted from list
**********Linked List operations*********
1.Insert        2.Delete        3.Display       4.Exit
Enter your choice: 3
List is empty
```

Experiment 8:

Consider a treasure hunt task where a series of clues are given. Clue1 given hint to clue2, clue2 provides hint for clue3 and so on until you can get a hint to the final treasure. Develop an illustration to demonstrate the above scenario.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *link;
};
typedef struct node *NODE;
NODE first=NULL;

NODE insert(NODE first)
{
   NODE temp;
   int item;
   printf("\nThe value to insert\n");
   scanf("%d",&item);
   temp=(NODE) malloc(sizeof(NODE));
   if(temp==NULL)
   {
      printf("out of memory\n");
      exit(0);
   }

   temp->info=item;
   temp->link=first;
   first=temp;
   printf("%d is inserted into list\n",temp->info);;
   return first;
}

NODE delete_sp(NODE first)
{
   NODE cur,prev;
   int key;
   printf("\nEnter the value to be deleted\n");
   scanf("%d",&key);
   if(first==NULL)
```

```
    {
       printf("List is underflow\n");
       return first;
    }
    /*if item to be deleted is in the first node*/
    if(key==first->info)
    {
       cur=first;
       first=first->link;
       printf("%d is deleted from list\n",cur->info);
       free(cur);
       return first;
    }
    /*if item to be deleted is in the middle/last node*/
    cur=first;
    prev=NULL;
    while(cur!=NULL && cur->info!=key)
    {
       prev=cur;
       cur=cur->link;
    }
    if(cur==NULL)
    {
       printf("Key not found\n");
       return first;
    }
    prev->link=cur->link;
    printf("%d is deleted from list\n",cur->info);
    free(cur);
    return first;
}

NODE display(NODE first)
{
    NODE cur;
    if(first==NULL)
    {
       printf("List is empty\n");
       return 0;
    }
    printf("Contents of the list are\n");
    cur=first;
```

```
   while(cur!=NULL)
   {
      printf("%d\t",cur->info);
      cur=cur->link;
   }
   printf("\n");
}

void main()
{
   int option,item;
   do
   {
      printf("\n**********Linked List operations*********\n");
      printf("1.Insert \t2.delete\t3.Display\t4.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&option);
      switch(option)
      {
         case 1:first=insert(first);
                break;
         case 2:first=delete_sp(first);
                break;
         case 3:display(first);
                break;
         case 4:exit(0);
      }
   }while(option!=4);
}
```

Output:

To Compile: **cc prog7.c**

To Execute: **./a.out**

```
**********Linked List operations*********
1.Insert   2.delete   3.Display   4.Exit
Enter your choice: 1
The value to insert
11
11 is inserted into list

**********Linked List operations*********
1.Insert   2.delete   3.Display   4.Exit
Enter your choice: 1
The value to insert
```

```
22
22 is inserted into list

**********Linked List operations*********
1.Insert   2.delete   3.Display   4.Exit
Enter your choice: 1
The value to insert
33
33 is inserted into list

**********Linked List operations*********
1.Insert   2.delete   3.Display   4.Exit
Enter your choice: 1

The value to insert
44
44 is inserted into list

**********Linked List operations*********
1.Insert   2.delete   3.Display   4.Exit
Enter your choice: 3
Contents of the list are
44   33   22   11


**********Linked List operations*********
1.Insert   2.delete   3.Display   4.Exit
Enter your choice: 2
Enter the value to be deleted
22
22 is deleted from list

**********Linked List operations*********
1.Insert   2.delete   3.Display   4.Exit
Enter your choice: 3
Contents of the list are
44   33   11
```

Experiment 9:

Consider a list of numbers. Find

i. Maximum number

ii. Minimum number

iii. Sum of all the numbers

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *link;
};
typedef struct node *NODE;
NODE first=NULL;

NODE insert(NODE first)
{
   NODE temp;
   int item;
   printf("\nThe value to insert\n");
   scanf("%d",&item);
   temp=(NODE) malloc(sizeof(NODE));
   if(temp==NULL)
   {
      printf("out of memory\n");
      exit(0);
   }

   temp->info=item;
   temp->link=first;
   first=temp;
   printf("%d is inserted into list\n",temp->info);;
   return first;
}
void display(NODE first)
{
   NODE cur;
   if(first==NULL)
   {
      printf("List is empty\n");
```

```
      exit(0);
   }
   printf("Contents of the list are\n");
   cur=first;
   while(cur!=NULL)
   {
      printf("%d\t",cur->info);
      cur=cur->link;
   }
   printf("\n");
}
void findMax(NODE first)
{
     int max;
     NODE ptr;
     if(first==NULL)
     {
          printf("List is empty\n");
          exit(0);
     }
     max=first->info;
     ptr=first;
     while(ptr!=NULL)
     {
          if(ptr->info>max)
          {
               max=ptr->info;
          }
          ptr=ptr->link;
     }
     printf("Maximum element of the list is %d\n",max);
}
void findMin(NODE first)
{
     int min;
     NODE ptr;
     if(first==NULL)
     {
          printf("List is empty\n");
          exit(0);
     }
     min=first->info;
```

```
     ptr=first;
     while(ptr!=NULL)
     {
          if(ptr->info<min)
          {
               min=ptr->info;
          }
          ptr=ptr->link;
     }
     printf("Minimum element of the list is %d\n",min);
}
void findSum(NODE first)
{
     int sum=0;
     NODE ptr;
     if(first==NULL)
     {
          printf("List is empty\n");
          exit(0);
     }
     ptr=first;
     while(ptr!=NULL)
     {
          sum=sum+ptr->info;
          ptr=ptr->link;
     }
     printf("Sum elements of the list is %d\n",sum);
}
void main()
{
   int option,item;
   do
   {
      printf("\n**********Linked List operations*********\n");
      printf("1.Insert
\t2.Display\t3.FindMax\t4.findMin\t5.findSum\t6.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&option);
      switch(option)
      {
         case 1:first=insert(first);
                break;
```

```
          case 2:display(first);
                 break;
          case 3:findMax(first);
                 break;
          case 4:findMin(first);
                     break;
             case 5: findSum(first);
                     break;
             case 6: exit(0);
       }
    }while(option!=6);
}
```

Output:

```
**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 1

The value to insert
45
45 is inserted into list

**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 1

The value to insert
65
65 is inserted into list

**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 1

The value to insert
12
12 is inserted into list

**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 1

The value to insert
```

```
78
78 is inserted into list

**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 1

The value to insert
64
64 is inserted into list

**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 2
Contents of the list are
64      78      12      65      45


**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 3
Maximum element of the list is 78

**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 4
Minimum element of the list is 12

**********Linked List operations*********
1.Insert        2.Display       3.FindMax       4.findMin
5.findSum       6.Exit
Enter your choice: 5
Sum elements of the list is 264
```

Experiment 10:

The phonebook will contain a list of contacts sorted in ascending order based on their names. Each contact will have a name and a phone number. Developing a phonebook management system using an ordered linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *link;
};
typedef struct node *NODE;
NODE first=NULL;

NODE insert(NODE first)
{
   NODE temp,cur,prev;
   int item;
   printf("\nEnter the value to be insert\n");
   scanf("%d",&item);

   /*obtain the new node*/
   temp=(NODE) malloc(sizeof(NODE));
   if(temp==NULL)
   {
      printf("out of memory\n");
      exit(0);
   }
   temp->info=item;
   temp->link=NULL;

   /*Inserting the node for the first time*/
   if(first==NULL)
   {
      first=temp;
      printf("%d is inserted into list\n",temp->info);
      return first;
   }

   /*inserting the node at the begining*/
   if(item < first->info)
```

```
    {
       temp->link=first;first=temp;
       printf("%d is inserted into list\n",temp->info);
       return first;
    }
    /* inserting item at the middle of the list*/
    /*find the cur and prev postions so that item can be inserted
      in between */
    cur=first;
    prev=cur;
    while(cur!=NULL && item>cur->info)
    {
       prev=cur;
       cur=cur->link;
    }
    prev->link=temp;
    temp->link=cur;
    printf("%d is inserted into list\n",temp->info);
    return first;
}

NODE display(NODE first)
{
    NODE cur;
    if(first==NULL)
    {
       printf("List is empty\n");
       return 0;
    }
    printf("Contents of the list are\n");
    cur=first;
    while(cur!=NULL)
    {
       printf("%d\t",cur->info);
       cur=cur->link;
    }
    printf("\n");
}

void main()
{
    int option,item;
```

```
    do
    {
       printf("\n****Ordered Linked List operations*****\n");
       printf("1.Insert\t2.Display\t3.Exit\n");
       printf("Enter your choice : ");
       scanf("%d",&option);

       switch(option)
       {
          case 1:first=insert(first);
                 break;
          case 2:display(first);
                 break;
          case 3:break;
          default:printf("Invalid option\n");
       }
    }while(option!=3);
}
```

Output:

To Compile: **cc prog8.c**

To Execute: **./a.out**

```
*****Ordered Linked List operations******
1.Insert   2.Display  3.Exit
Enter your choice : 1
Enter the value to be insert
11
11 is inserted into list

*****Ordered Linked List operations******
1.Insert   2.Display  3.Exit
Enter your choice : 1
Enter the value to be insert
22
22 is inserted into list

*****Ordered Linked List operations******
1.Insert   2.Display  3.Exit
Enter your choice : 1
Enter the value to be insert
55
55 is inserted into list

*****Ordered Linked List operations******
1.Insert   2.Display  3.Exit
```

```
Enter your choice : 1
Enter the value to be insert
33
33 is inserted into list

*****Ordered Linked List operations******
1.Insert   2.Display  3.Exit
Enter your choice : 1
Enter the value to be insert
25
25 is inserted into list
*****Ordered Linked List operations******
1.Insert   2.Display  3.Exit
Enter your choice : 1
Enter the value to be insert
67
67 is inserted into list

*****Ordered Linked List operations******
1.Insert   2.Display  3.Exit
Enter your choice : 2
Contents of the list are
11   22   25   33   55   67
```

Experiment 11:

Assume you have an ipod, where in you have stored plenty of songs so that you get engaged during a long journey. If you want to hear a particular song, you need to use forward button to reach that song and can also traverse back using backward button. Implement the following using relevant data structure.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *llink;
   struct node *rlink;
};
typedef struct node * NODE;
NODE first=NULL;

NODE insert_begin(NODE first)
{
   NODE temp;
   int item;
   printf("enter the value\n");
   scanf("%d",&item);
   temp=(NODE) malloc(sizeof(NODE));
   if(temp==NULL)
   {
      printf("out of memory n");
      exit(0);
   }
   temp->info=item;
   temp->rlink=NULL;
   temp->llink=NULL;
   if(first==NULL)
   {
          first=temp;
          printf("%d is inserted\n",temp->info);
          return first;
   }
   first->llink=temp;
   temp->rlink=first;
   printf("%d is inserted\n",temp->info);
```

```
    first=temp;
    return first;
}

NODE delete_begin(NODE first)
{
    NODE cur;
    if(first==NULL)
    {
       printf("list is empty\n");
       return first;
    }
    cur=first;
    first=first->rlink;
    if(first!=NULL)
     first->llink=NULL;
    printf("%d is deleted\n",cur->info);
    free(cur);
    return first;
}

NODE display(NODE first)
{
    NODE cur;
    if(first==NULL)
    {
       printf("list is empty\n");
       return first;
    }
    cur=first;
    printf("contents of the list are\n");
    while(cur!=NULL)
    {
       printf("%d\t",cur->info);
       cur=cur->rlink;
    }
    printf("\n");
    return first;
}
int main()
{
    int option;
```

```
    do{
       printf("\n doubly linked list operations\n");
       printf("1: insert\t2: delete\t3: display\t4:exit\n");
       printf("enter the option:");
       scanf("%d",&option);
       switch(option)
       {
          case 1:first=insert_begin(first);
                 break;
          case 2:first=delete_begin(first);
                 break;
          case 3:first=display(first);
                 break;
          case 4:break
       }
    }while(option!=4);
}
```

Output:

To Compile: **cc prog9.c**

To Execute: **./a.out**

```
doubly linked list operations
1: insert  2: delete  3: display  4:exit
enter the option: 1
enter the value
11
11 is inserted

doubly linked list operations
1: insert  2: delete  3: display  4:exit
enter the option: 1
enter the value
33
33 is inserted

doubly linked list operations
1: insert  2: delete  3: display  4:exit
enter the option: 1
enter the value
55
55 is inserted

doubly linked list operations
1: insert  2: delete  3: display  4:exit
enter the option: 3
contents of the list are
55   33   11

doubly linked list operations
```

```
1: insert  2: delete  3: display  4:exit
enter the option: 2
55 is deleted

doubly linked list operations
1: insert  2: delete  3: display  4:exit
enter the option: 2
33 is deleted

doubly linked list operations
1: insert  2: delete  3: display  4:exit
enter the option: 3
contents of the list are
11
```

Experiment 12:

Your text book contains chapters, sections, subsections, subdivisions, etc. Illustrate this scenario of text book using tree structure.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int info;
   struct node *llink;
   struct node *rlink;
};
typedef struct node *NODE;
NODE root=NULL;

NODE insert(NODE root)
{
   NODE temp,cur,prev;
   int item;
   printf("enter the value\n");
   scanf("%d",&item);
   temp=(NODE) malloc(sizeof(NODE));
   if(temp==NULL)
   {
      printf("out of memory n");
      exit(0);
   }

   temp->info=item;
   temp->llink=NULL;
   temp->rlink=NULL;

   if(root==NULL)
   {
      root=temp;
      printf("%d is inserted\n",temp->info);
      return root;
   }

   prev=NULL;
   cur=root;
   while(cur!=NULL)
```

```
    {
       prev=cur;
       if(item<cur->info)
          cur=cur->llink;
       else
          cur=cur->rlink;
    }
    if(item<prev->info)
    {
       prev->llink=temp;
       printf("%d is inserted as left child of %d\n",temp->info,
            prev->info);
    }
    else
    {
       prev->rlink=temp;
       printf("%d is inserted as right child of %d\n",temp->info,
       prev->info);
    }
    return root;
}

void preorder(NODE root)
{
    if(root==NULL)
       return;
    printf("%d\t",root->info);
    preorder(root->llink);
    preorder(root->rlink);
}

void inorder(NODE root)
{
    if(root==NULL)
       return;
    inorder(root->llink);
    printf("%d\t",root->info);
    inorder(root->rlink);
}

void postorder(NODE root)
{
```

```
    if(root==NULL)
       return;
    postorder(root->llink);
    postorder(root->rlink);
    printf("%d\t",root->info);
}


int main()
{
    int n;
    printf("Enter the number of node\n");
    scanf("%d",&n);
    while(n>0)
    {
       root=insert(root);
       n=n-1;
    }
    if(root==NULL)
    {
       printf("Tree is empty\n");
       exit(0);
    }
    printf("\nPreorder traversal of tree\n");
    preorder(root);
    printf("\nInorder traversal of tree\n");
    inorder(root);
    printf("\nPostorder traversal of tree\n");
    postorder(root);
}
```

Output:

To Compile: **cc prog10.c**

To Execute: **./a.out**

```
Enter the number of node
4
enter the value
11
11 is inserted
enter the value
55
55 is inserted as right child of 11
enter the value
23
```

```
23 is inserted as left child of 55
enter the value
43
43 is inserted as right child of 23

Preorder traversal of tree
11   55   23   43
Inorder traversal of tree
11   23   43   55
Postorder traversal of tree
43   23   55   11
```