# C Programming & Data Structures LAB (20A05201P)

## SAMPLE RECORD

## I - B. TECH & I- SEM

Prepared by:

**P Khatija Khan, Assistant Professor K Kushboo, Assistant Professor**

**Department of Computer Science & Engineering**C Programming & Data Structures Lab

VEMU INSTITUTE OF TECHNOLOGY

(Approved By AICTE, New Delhi and Affiliated to JNTUA, Anantapur) Accredited By NAAC, NBA (EEE, ECE & CSE) & ISO: 9001-2015 Certified Institution

Near Pakala, Kothakota, Chittoor- Tirupati Highway Chittoor, Andhra Pradesh-517 112

**Week-1**

**Write C programs that use both recursive and non-recursive functions**

i. **To find the factorial of a given integer.**
ii. **To find the GCD (Greatest Common Divisor) of two given integers.**
iii. **To solve Towers of Hanoi Problem.**

(1) ***Factorial of a given number by using Recursive function***

***Algorithm:-***

1. Start
2. Read a number N
3. Call a function factorial(N) by passing the values of N
4. If N=1 then it returns 1 as the factorial
5.Otherwise, it calculates the factorial f = N * factorial(N-1) by calling the same function again and again
6. Display the factorial of number N
7. Stop

***Program#1(a) Factorial of a given number by using Recursive function***

```
#include<stdio.h>
int fact(int n)
{
if(n==0)
return 1;
else
return n*fact(n-1);
}
void main()
```



```
{
int n,f;
printf("\n enter a no.: ");
scanf("%d",&n);  f=fact(n);
printf("\n factorial of %d is %d",n,f);
}
```

**Factorial of a given number by using Non-Recursive function**

***Algorithm***

1. Start
2. Read a number N
3. Initialize fact =1

4. Calculate fact = fact * N using a loop and decrement N value till N =1
5. Display the factorial of number N
6. Stop

**Program#1(b) Factorial of a given number by using Non-Recursive function**

```
#include<stdio.h>
int fact(int n)
{
int f=1,i;
if((n==0)||(n==1))
return(1);
else
{
for(i=1;i<=n;i++)
f=f*i;
}
return(f);
}
void main()
{
int n;
printf("enter the number :");
scanf("%d",&n);
printf("factoria of number%d",fact(n));
}
```

---

***(ii)* To find the GCD (Greatest Common Divisor) of two given integers.**

***GCD of a given two integer by using Recursive Function***

***<u>Algorithm</u>:-***

1. Start
2. Read two numbers a and b
3. Call a function gcd (a, b) by passing the value of a and b
4. Check a != b, then
5. Check a > b then calculate gcd of two numbers by calling the function gcd(a-b, b)
6. Otherwise calculate gcd of two numbers by calling the function gcd(a, b-a)
7. Display the gcd of two numbers
8. Stop

***Program#2(a) GCD of a given two integer by using Recursive Function***

```
#include<stdio.h>
int gcd(int m,int n)
{
if(n==0)
return m;
else
gcd(n,m%n);
}
void main()
{
int a,b,g,l;
printf("\n enter a: ");
scanf("%d",&a);
printf("\n enter b: ");
scanf("%d",&b);
g=gcd(a,b);
l=(a*b)/g;
printf("\n GCD of %d and %d : %d",a,b,g);
printf("\n LCM of %d and %d : %d",a,b,l);
}
```

**Program#2(b) GCD of a given two integer by using Non-Recursive Function**

***Algorithm*:-**

1. Start
2. Read two numbers a and b
3. Check if a != b then
4. Check if a >= b - 1 then set a = a – b otherwise set b = b – a
5. Display the gcd as the value of a.
6. Stop

***Program#2(b) : GCD of a given two integer by using Non-Recursive Function***

```
#include<stdio.h>
int gcd(int,int);
void main()
{
int a,b,x;
printf("\n enter a: ");
scanf("%d",&a);
printf("\n enter b: ");
scanf("%d",&b);;
x=gcd(a,b);
printf("G.C.D of %d and %d is %d",a,b,x);
}
int gcd(int a,int b)
```

```
{
int r;
while(b!=0)
{
r=a%b;
a=b;
b=r;
}
return a;
}
```

***(iii) To solve Towers of Hanoi Problem***

**Program#3(a)** ***Towers of Hanoi problem using Non-Recursive function***

```
#include<stdio.h>
void Hanoinonrecursion(int num,char sndl,char indl,char dndl)
{
char stkn[100],stksndl[100],stkindl[100],stkdndl[100],stkadd[100],temp;
int top,add;
top=NULL;
one:
if(num==1)
{
printf("\n Move top disk from needle %c to needle %c",sndl,dndl);
goto four;
}
two:
top=top+1;
stkn[top]=num;
stksndl[top]=sndl;
stkindl[top]=indl;
stkdndl[top]=dndl;
stkadd[top]=3;
num=num-1;
sndl=sndl;
temp=indl;
indl=dndl;
dndl=temp;
goto one;
```

```
three:
printf("\n Move top disk from needle %c to needle %c",sndl,dndl);
top=top+1;
stkn[top]=num;
stksndl[top]=sndl;
stkindl[top]=indl;
stkdndl[top]=dndl;
stkadd[top]=5;
num=num-1;
temp=sndl;
sndl=indl;
indl=temp;
dndl=dndl;
goto one;
four:
if(top==NULL)
return;
num=stkn[top];
sndl=stksndl[top];
indl=stkindl[top];
dndl=stkdndl[top];
add=stkadd[top];
top=top-1;
if(add==3)
goto three;
else if(add==5)
goto four;
}
void main()
{
int no;
printf("Enter the no. of diss to be transferred:");
scanf("%d",&no);
if(no<1)
printf("\n There's nothing to move");
else
printf("\n nonrecursive");
Hanoinonrecursion(no,'A','B','C');
}
```

***Program#3(b):To solve Towers of Hanoi problem using Recursive function***

```
#include<stdio.h>
void Hanoirecursion(int num,char ndl1,char ndl2,char ndl3)
{
```

```
if(num==1)
{
printf("Move top disk from needle %c to needle %c",ndl1,ndl2);
return;
}
Hanoirecursion(num-1,ndl1,ndl3,ndl2);
printf("Move top dis from needle %c to needlle %c",ndl1,ndl2);
Hanoirecursion(num-1,ndl3,ndl2,ndl1);
}
void main()
{
int no;
printf("Enter the no. of disk to be transferred:");
scanf("%d",&no);
if(no<1)
printf("\n There's nothing to move");
else
printf("\n recursive");
Hanoirecursion(no,'A','B','C');
}
```

**Week-2**

**(1) Write a C program to find out the largest and smallest number in a list of integers**

**(2) Write a C program that uses functions to perform the following:**
**i) Addition of Two Matrices ii) Multiplication of Two Matrices**

**(1) Write a C program to find out the largest and smallest number in a list of integers**

**Problem**

Let the user enter four series of integers in the console, find out a number which is smallest and largest in a series

**Solution**

To calculate the small and large number, we use if conditions. The logic we use to find the largest and
smallest number is –

```
if(minno>q) //checking 1st and 2nd number
  minno=q;
else if(maxno&l;q)
  maxno=q;
if(minno>r) //checking 1st and 3rd number
  minno=r;
```

***Program#1: program to find out the largest and smallest number in a list of integers***

```
#include<stdio.h>
int main(){
  int minno,maxno,p,q,r,s;
  printf("enter any four numbers:");
  scanf("%d%d%d%d",&p,&q,&r,&s);
  minno=p;
  maxno=p;
  if(minno>q) //checking 1st and 2nd number
    minno=q;
  else if(maxno<q)
    maxno=q;
  if(minno>r) //checking 1st and 3rd number
    minno=r;
  else if(maxno<r)
    maxno=r;
  if(minno>s) //checking 1st and 4th number
    minno=s;
  else if(maxno<s)
    maxno=s;
  printf("Largest number from the given 4 numbers is:%d\n",maxno);
  printf("Smallest numbers from the given 4 numbers is:%d",minno);
  return 0;
}
```

***Output***

```
enter any four numbers:34 78 23 12
Largest number from the given 4 numbers is:78
Smallest numbers from the given 4 numbers is:12
```

***Program#2: Program finds the smallest and largest element in an array***

```
#include<stdio.h>
int main(){
  int a[50],i,num,large,small;
  printf("Enter the number of elements :");
  scanf("%d",&num);
  printf("Input the array elements :\n");
  for(i=0;i<num;++i)
    scanf("%d",&a[i]);
  large=small=a[0];
  for(i=1;i<num;++i){
    if(a[i]>large)
      large=a[i];
    if(a[i]<small)
      small=a[i];
  }
```

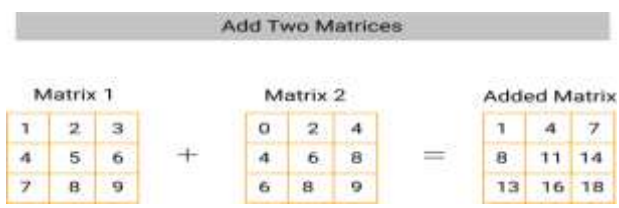
```
  printf("small=  %d\n",small);
  printf("large=   %d\n",large);
  return 0;
}
```

**Output**

```
Enter the number of elements :8
Input the array elements :
1
2
6
4
8
9
3
9
small= 1
large= 9
```

---

**(2) Write a C program that uses functions to perform the following:**
**i) Addition of Two Matrices ii) Multiplication of Two Matrices**




**2(i) Algorithm:**

Step 1: Start
Step 2: Declare i, j, k, A[3][3], B[3][2], C[3][2] as integers
Step 3: Initialize i = 0, j = 0
Step 4: Till i < 3 execute step 5 else goto step 9
Step 5: Till j < 3 execute steps 6 to 7 else goto step 8
Step 6: Read A[i][j]
Step 7: Increment j by 1 goto step 5
Step 8: Increment i by 1 goto step 4
Step 9: Initialize i = 0, j = 0
Step 10: Till i < 3 execute step 11 else goto step15
Step 11: Till j < 2 execute steps 6 to 7 else goto step 14
Step 12: Read B[i][j]

Step 13: Increment j by 1 goto step 11
Step 14: Increment i by 1 goto step 10
Step 15: Initialize i = 0, j = 0, k = 0
Step 16: Till i < 3 execute step 17 else goto step 24
Step 17: Till j < 2 execute step 18 else goto step 23
Step 18: Initialize C[i][j] = 0
Step 19: Till k<3 execute steps 20 to 21 else goto step
Step 20: calculate C[i][j] = C[i][j] + A[i][k] * B[k][j]
Step 21: Increment k by 1 goto step 19
Step 22: Increment j by 1 goto step 17
Step 23: Increment i by 1 goto step 16
Step 24: Initialize i = 0, j = 0
Step 25: Till i < 3 execute step 26 else goto step 30
Step 26: Till j < 3 execute steps 27 to 28 else goto step 29
Step 27: Print C[i][j]
Step 28: Increment j by1 goto step 26
Step 29: Increment i by 1 goto step 25
Step 30: Stop

**Addition of Two Matrices**

**Program:**

```
#include   <stdio.h>
#include  <conio.h>
void main()
{
  int a[3][3], b[3][3], c[3][3], i, j;
  clrscr();
  printf("Enter the elements of 3*3 matrix a \n");
  for(i = 0; i < 3; i++)
  {
    for(j = 0; j < 3; j++)
    {
   scanf("%d", &a[i][j]);
    }
  }
  printf("Enter the elements of 3*3 matrix b \n");
  for(i = 0; i < 3; i++)
  {
    for(j = 0; j < 3; j++)
    {
  scanf("%d", &b[i][j]);
    }
```

```
  }
  for(i = 0; i < 3; i++)
  {
    for(j = 0; j < 3; j++)
    {
   c[i][j] = a[i][j] + b[i][j];
    }
  }
  printf("The resultant 3*3 matrix c is \n");
  for(i = 0; i < 3; i++)
  {
    for(j = 0; j < 3; j++)
    {
   printf("%d\t", c[i][j]);
    }
    printf("\n");
  }
  getch();
}
```

**Input & Output:**

```
Enter the elements of 3*3 matrix a
1 2 3 4 5 6 7 8 9
Enter the elements of 3*3 matrix  b
1 2 3 4 5 6 7 8 9
The resultant 3*3 matrix  c is
2      4      6
8     10     12
14    16     18
```

**2(ii) Multiplication of Two Matrices**

**Program:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a[3][3], b[3][3], c[3][3], i, j, k;
  clrscr();
  printf("Enter the elements of 3*3 matrix a \n");
  for(i = 0; i < 3; i++)
  {
```

```
    for(j = 0; j < 3; j++)
    {
   scanf("%d", &a[i][j]);
    }
  }
  printf("Enter the elements of 3*3 matrix b \n");
  for(i = 0; i < 3; i++)
  {
    for(j = 0; j < 3; j++)
    {
   scanf("%d", &b[i][j]);
    }
  }
  for(i = 0; i < 3; i++)
  {
    for(j = 0; j < 3; j++)
    {
   c[i][j] = 0
   for(k = 0; k < 3; k++)
   {
     c[i][j] = c[i][j] + (a[i][k] * b[k][j])
   }
    }
  }
  printf("The resultant 3*3 matrix c is \n");
  for(i = 0; i < 3; i++)
  {
    for(j = 0; j < 3; j++)
    {
   printf("%d\t", c[i][j]);
    }
    printf("\n");
  }
  getch();
}
```

**Input & Output:**

```
Enter the elements of 3*3 matrix a
1 2 3 4 5 6 7 8 9
Enter the elements of 3*3 matrix  b
1 2 3 4 5 6 7 8 9
The resultant 3*3 matrix c is
30    36      42
```

55 81 96
102 126 150

**Week-3**

**Write a C program that uses functions to perform the following operations:**

i. **To insert a sub-string in to a given main string from a given position.**
ii. **To delete n Characters from a given position in a given string.**

**i) To insert a sub-string in to a given main string from a given position.**

**Algorithm:**

Step 1: Start
Step 2: read main string and sub string
Step 3: find the length of main string(r)
Step 4: find length of sub string(n)
Step 5: copy main string into sub string
Step 6: read the position to insert the sub string(p)
Step 7: copy sub string into main string from position p - 1
Step 8: copy temporary string into main string from position p + n - 1
Step 9: print the strings
Step 10: Stop

**Program:**

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
  char str1[20], str2[20];
  int l1, l2, n, i;
  clrscr();
  puts("Enter the string 1\n");
  gets(str1);
  l1 = strlen(str1);
  puts("Enter the string 2\n");
  gets(str2);
  l2 = strlen(str2);
  printf("Enter the position where the string is to be inserted\n");
  scanf("%d", &n);
```

```
  for(i = n; i < l1; i++)
 {
    str1[i + l2] = str1[i];
  }
  for(i = 0; i < l2; i++)
  {
    str1[n + i] = str2[i];
  }
    str2[l2 + 1] = '\0';
  printf("After inserting the string is %s", str1);
  getch();
}
```

**Input & Output:**

```
Enter the string 1
sachin
Enter the string 2
tendulkar
Enter the position where the string is to be inserted
4
After inserting the string is  sachtendulkarin
```

**ii) To delete n Characters from a given position in a given string.**

**Algorithm:**

Step 1: Start
Step 2: read string
Step 3: find the length of the string
Step 4: read the value of number of characters to be deleted and positioned
Step 5: string copy part of string from position to end, and (position + number of characters to end)
Step 6: Stop

**Program:**

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
  char str[20];
  int i, n, l, pos;
```

```
  clrscr();
  puts("Enter the string\n");
  gets(str);
  printf("Enter the position where the characters are to be deleted\n");
  scanf("%d", &pos);
  printf("Enter the number of characters to be deleted\n");
  scanf("%d", &n);
  l = strlen(str);
  for(i = pos + n; i < l; i++)
  {
    str[i - n] = str[i];
  }
  str[i - n] = '\0';
  printf("The string is %s", str);
  getch();
}
```

**Input & Output:**

```
Enter the string
sachin
Enter the position where characters are to be deleted
2
Enter the number of characters to be deleted
2
The string is  sain
```

**Week-4**

**(1) Write a C program that displays the position or index in the string S where the string T begins, or – 1 if S doesn't contain T.**

**Algorithm:**

Step 1: Start
Step 2: read the string and then displayed
Step 3: read the string to be searched and then displayed
Step 4: searching the string T in string S and then perform the following steps
i. found = strstr(S, T)
ii. if found print the second string is found in the first string at the position. If not goto step5
Step 5: print the -1
Step 6: Stop

**Program:**

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void main()
{
  char s[30], t[20];
  char *found;
  clrscr();
  puts("Enter the first string: ");
  gets(s);
  puts("Enter the string to be searched: ");
  gets(t);
  found = strstr(s, t);
  if(found)
  {
    printf("Second String is found in the First String at %d position.\n", found - s);
  }
  else
  {
    printf("-1");
  }
  getch();
}
```

**Input & Output:**

1. Enter the first string:
Vemu Institute
Enter the string to be searched:
mu
second string is found in the first string at 2 position

2. Enter the first string:
nagaraju
Enter the string to be searched:
raju
second string is found in the first string at 4 position
3.Enter the first string:
nagarjuna
Enter the string to be searched:
ma
-1

**Write a C program to count the lines, words and characters in a given text.**

**Algorithm:**

Step 1: Start
Step 2: Read the text until an empty line
Step 3: Compare each character with newline char '\n' to count no of lines
Step 4: Compare each character with tab char '\t\' or space char ' ' to count no of words
Step 5: Compare first character with NULL char '\0' to find the end of text
Step 6: No of characters = length of each line of text
Step 7: Print no of lines, no of words, no of chars
Step 8: Stop

**Program:**

```
#include   <stdio.h>
#include  <conio.h>
#include  <string.h>
void main()
{
  char str[100];
  int i = 0, l = 0, f = 1;
  clrscr();
  puts("Enter any string\n");
  gets(str);
  for(i = 0; str[i] !='\0'; i++)
  {
    l = l + 1;
  }
  printf("The number of characters in the string are %d\n", l);
  for(i = 0; i <= l-1; i++)
  {
    if(str[i] == ' ')
    {
   f = f + 1;
    }
  }
  printf("The number of words in the string are %d", f);
  getch();
}
```

**Input & Output:**

Enter any string

abc def ghi jkl mno pqr stu vwx yz
The number of characters in the string are 34
The number of words in the string are 9

**Week-5**

(1) **C Program to Perform All Arithmetic Operations using Pointers**
(2) **Write a C Program to demonstrate the following parameter passing mechanisms: i) call-by-value ii) call-by-reference**

**Input from user:**

Enter Number1: 20

Enter Number2: 10

**Expected output:**

Sum=30

Subtraction=10

Multiplication=200

Division=2.0000

Before learning how to write a C program to perform all arithmetic operations we need to know what is arithmetic operators means.

**What is Arithmetic Operators:**

Arithmetic Operators are the operators which are used to perform various mathematical operations such as addition, subtraction, division, multiplication etc.

**1. Plus (+):** is used for addition like 5 + 5 = 10, 35 + 35 = 70, 10 + 13 + 25 = 48 etc.

**2. Minus Operator (-):** is used for subtract one number from another number like 23 - 8 = 15, 65 - 25 = 40.

**3.Multiplication Operator (*) :** is represented as an asterisk (*) symbol and it returns product of given numbers like 5 * 5 = 25, 7 * 12 = 84 etc.

**4.Division Operator (/) :** is denoted by forward slash (/) symbol and it divides first number by the second number like 35 / 5 = 7.

5. **Modulus Operator (%) :** is denoted by percentage (%) symbol and it returns remainder. for ex: 25 / 5 = 0, 45 / 6 = 3 etc.

**C Program to Perform All Arithmetic Operations using Pointers:**

```
#include<stdio.h>
int main()
{
        int  no1,no2;
        int *ptr1,*ptr2;
     int sum,sub,mult;
     float div;
   printf("Enter number1:\n");
   scanf("%d",&no1);
   printf("Enter number2:\n");
   scanf("%d",&no2);

   ptr1=&no1;//ptr1 stores address of no1
   ptr2=&no2;//ptr2 stores address of no2

   sum=(*ptr1) + (*ptr2);
   sub=(*ptr1) - (*ptr2);
   mult=(*ptr1) * (*ptr2);
   div=(*ptr1) / (*ptr2);

   printf("sum= %d\n",sum);
   printf("subtraction= %d\n",sub);
   printf("Multiplication= %d\n",mult);
   printf("Division= %f\n",div);

   return 0;
}
```

**Step by step logic of the given program:**

1. Accept two numbers from user store it in variable say no1 and no2.

2. Store address of no1 in pointer variable ptr1 and store address of no2 in variable ptr2 using reference operator (&):

```
ptr1=&no1;
ptr2=&no2;
```

3. After that calculate addition, subtraction, multiplication and division using dereference operator(*):

```
sum=(*ptr1) + (*ptr2);
sub=(*ptr1) - (*ptr2);
mult=(*ptr1) * (*ptr2);
div=(*ptr1) / (*ptr2);
```

4. Last print sum, sub, mult and div on the output screen.

***Call by Value***

```
#include<stdio.h>
void change(int num) {
   printf("Before adding value inside function num=%d \n",num);
   num=num+100;
   printf("After adding value inside function num=%d \n", num);
}
int main() {
   int x=100;
   printf("Before function call x=%d \n", x);
   change(x);//passing value in function
   printf("After function call x=%d \n", x);
return 0;
}
```

**Output**

```
Before function call x=100
Before adding value inside function num=100
After adding value inside function num=200
After function call x=100
```

***Call by Reference***

```
#include<stdio.h>
void change(int *num) {
   printf("Before adding value inside function num=%d \n",*num);
   (*num) += 100;
```

```
   printf("After adding value inside function num=%d \n", *num);
}
int main() {
   int x=100;
   printf("Before function call x=%d \n", x);
   change(&x);//passing reference in function
   printf("After function call x=%d \n", x);
return 0;
}
```

**Output**

```
Before function call x=100
Before adding value inside function num=100
After adding value inside function num=200
After function call x=200
```

---

***Call by Value - Swapping of Two Numbers***

```
#include <stdio.h>
void swap(int , int); //prototype of the function
int main()
{
   int a = 10;
   int b = 20;
   printf("Before swapping the values in main a = %d, b = %d\n",a,b); // printing the value of a a
nd b in main
   swap(a,b);
   printf("After swapping values in main a = %d, b = %d\n",a,b); // The value of actual parameter
s do not change by changing the formal parameters in call by value, a = 10, b = 20
}
void swap (int a, int b)
{
   int temp;
   temp = a;
   a=b;
   b=temp;
   printf("After swapping values in function a = %d, b = %d\n",a,b); // Formal parameters, a = 20
, b = 10
}
```

**Output**

Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 10, b = 20

***Call by Reference – Swapping of Two Numbers***

```
#include <stdio.h>
void swap(int *, int *); //prototype of the function
int main()
{
   int a = 10;
   int b = 20;
   printf("Before swapping the values in main a = %d, b = %d\n",a,b);
 // printing the value of a and b in main
   swap(&a,&b);
   printf("After swapping values in main a = %d, b = %d\n",a,b);
// The values of actual parameters do change in call by reference, a = 10, b = 20
}
void swap (int *a, int *b)
{
   int temp;
   temp = *a;
   *a=*b;
   *b=temp;
   printf("After swapping values in function a = %d, b = %d\n",*a,*b);
// Formal parameters, a = 20, b = 10
}
```

**Output**

Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 20, b = 10

**Week-6**

**Write a C program to perform i)Reading a complex number ii) Writing a complex number iii) Addition of two complex numbers iv) Multiplication of two complex numbers (Note: represent complex number using a structure.) using Functions**

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

```
#include <conio.h>
struct complex
{
  float real, imag;
}a, b, c;
  struct complex read(void);
  void write(struct complex);
  struct complex add(struct complex, struct complex);
  struct complex sub(struct complex, struct complex);
  struct complex mul(struct complex, struct complex);
  struct complex div(struct complex, struct complex);
void main ()
{
  clrscr();
  printf("Enter the 1st complex number\n ");
  a = read();
  write(a);
  printf("Enter the 2nd complex number\n");
  b = read();
  write(b);
  printf("Addition\n ");
  c = add(a, b);
  write(c);
  printf("Substraction\n ");
  c = sub(a, b);
  write(c);
  printf("Multiplication\n");
  c = mul(a, b);
  write(c);
  printf("Division\n");
  c = div(a, b);
 write(c);
 getch();
}
struct complex read(void)
{
  struct complex t;
  printf("Enter the real part\n");
  scanf("%f", &t.real);
  printf("Enter the imaginary part\n");
  scanf("%f", &t.imag);
  return t;
}
void write(struct complex a)
```

```
{
   printf("Complex number is\n");
   printf(" %.1f + i %.1f", a.real, a.imag);
   printf("\n");
}
struct complex add(struct complex p, struct complex q)
{
   struct complex t;
   t.real = (p.real + q.real);
   t.imag = (p.imag + q.imag);
   return t;
}
struct complex sub(struct complex p, struct complex q)
{
   struct complex t;
   t.real = (p.real - q.real);
   t.imag = (p.imag - q.imag);
   return t;
}
struct complex mul(struct complex p, struct complex q)
{
   struct complex t;
   t.real=(p.real * q.real) - (p.imag * q.imag);
   t.imag=(p.real * q.imag) + (p.imag * q.real);
   return t;
}
struct complex div(struct complex p, struct complex q)
{
   struct complex t;
   t.real = ((p.imag * q.real) - (p.real * q.imag)) / ((q.real * q.real) + (q.imag * q.imag));
   t.imag = ((p.real * q.real) + (p.imag * q.imag)) / ((q.real * q.real) + (q.imag * q.imag));
   return(t);
}
```

**Input & Output:**

```
Enter the real part
2
Enter the imaginary part
4
Complex number is
2.0 + i4.0
Enter the real part
4
```

```
Enter the imaginary part
2
Complex number is
4.0 + i2.0
Addition
Complex number is
6.0 + i6.0
Subtraction
Complex number is
-2.0 + i2.0
Multiplication
Complex number is
0.0 + i20.0
Division
Complex number is
0.6 + i0.8
```

---

Week 7

Write C programs that implement stack (its operations) using

i) Arrays

ii) Pointers

```
/*  C Program to implement Stack Operations Using Pointer  */

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

#define MAX 50
int size;

// Defining the stack structure
struct stack {
   int arr[MAX];
   int top;
};

// Initializing the stack(i.e., top=-1)
void init_stk(struct stack *st) {
   st->top = -1;
}
```

```
// Entering the elements into stack
void push(struct stack *st, int num) {
  if (st->top == size - 1) {
    printf("\nStack overflow(i.e., stack full).");
    return;
  }
  st->top++;
  st->arr[st->top] = num;
}

//Deleting an element from the stack.
int pop(struct stack *st) {
  int num;
  if (st->top == -1) {
    printf("\nStack underflow(i.e., stack empty).");
    return NULL;
  }
  num = st->arr[st->top];
  st->top--;
  return num;
}

void display(struct stack *st) {
  int i;
  for (i = st->top; i >= 0; i--)
    printf("\n%d", st->arr[i]);
}

int main()
{
  int element, opt, val;
  struct stack ptr;
  init_stk(&ptr);
  printf("Enter Stack Size :: ");
  scanf("%d", &size);
  while (1) {
    printf("\n\tSTACK PRIMITIVE OPERATIONS. .......\n");
    printf("\n1.PUSH");
    printf("\n2.POP");
    printf("\n3.DISPLAY");
    printf("\n4.QUIT");
    printf("\n");
    printf("\nEnter your option :: ");
```

```
    scanf("%d", &opt);
    switch (opt)
    {
    case 1:
      printf("\nEnter the element into stack:");
      scanf("%d", &val);
      push(&ptr, val);
      break;

    case 2:
      element = pop(&ptr);
      printf("\nThe element popped from stack is : %d", element);
      break;

    case 3:
      printf("\nThe current stack elements are:");
      display(&ptr);
      break;

    case 4:
      exit(0);

    default:
      printf("\nEnter correct option!Try again.");

    }
  }
  return (0);
}
```

**Output:**

```
/*  C Program to implement Stack Operations Using Pointer  */

Enter Stack Size :: 6

      STACK PRIMITIVE OPERATIONS.........

1.PUSH
```

2. POP
3. DISPLAY
4. QUIT

Enter your option :: 1

Enter the element into stack:1

STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 1

Enter the element into stack:2

STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 1

Enter the element into stack:3

STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 1

Enter the element into stack:4

STACK PRIMITIVE OPERATIONS.....

1.PUSH

2. POP
3. DISPLAY
4. QUIT

Enter your option :: 1

Enter the element into stack:5

STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 1

Enter the element into stack:6

STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 1

Enter the element into stack:7

Stack overflow(i.e., stack full).

STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 3

The current stack elements are:
6
5

4

```
3
2
1
        STACK PRIMITIVE OPERATIONS.........

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

The element popped from stack is : 6
        STACK PRIMITIVE OPERATIONS.........

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

The element popped from stack is : 5
        STACK PRIMITIVE OPERATIONS.........

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

The element popped from stack is : 4
        STACK PRIMITIVE OPERATIONS.........

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

The element popped from stack is : 3

        STACK PRIMITIVE OPERATIONS.........
```

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

The element popped from stack is : 2
STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

The element popped from stack is : 1
STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

Stack underflow(i.e., stack empty).
The element popped from stack is : 0
STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 2

Stack underflow(i.e., stack empty).
The element popped from stack is : 0
STACK PRIMITIVE OPERATIONS.....

1. PUSH

2. POP
3. DISPLAY
4. QUIT

Enter your option :: 3

The current stack elements are:
STACK PRIMITIVE OPERATIONS.....

1. PUSH
2. POP
3. DISPLAY
4. QUIT

Enter your option :: 4

Process returned 0

---

### C Program for STACK Using Arrays

```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{
   //clrscr();
   top=-1;
   printf("\n Enter the size of STACK[MAX=100]:");
   scanf("%d",&n);
   printf("\n\t STACK OPERATIONS USING ARRAY");
   printf("\n\t--------------------------------");
   printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
   do
   {
      printf("\n Enter the Choice:");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         {
            push();
            break;
         }
```

```
            case 2:
            {
               pop();
               break;
            }
            case 3:
            {
               display();
               break;
            }
            case 4:
            {
               printf("\n\t EXIT POINT ");
               break;
            }
            default:
            {
               printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
            }

         }
      }
      while(choice!=4);
      return 0;
   }
   void push()
   {
      if(top>=n-1)
      {
         printf("\n\tSTACK is over flow");

      }
      else
      {
         printf(" Enter a value to be pushed:");
         scanf("%d",&x);
         top++;
         stack[top]=x;
      }
   }
   void pop()
   {
      if(top<=-1)
      {
```

```
        printf("\n\t Stack is under flow");
    }
    else
    {
        printf("\n\t The popped elements is %d",stack[top]);
        top--;
    }
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }

}
```

Output:

```
Enter the size of STACK[MAX=100]:10

        STACK OPERATIONS USING ARRAY
       ----------------------------------------
        1. PUSH
        2. POP
        3. DISPLAY
        4. EXIT
 Enter the Choice:1
 Enter a value to be pushed:12

 Enter the Choice:1
 Enter a value to be pushed:24

 Enter the Choice:1
 Enter a value to be pushed:98
```

```
Enter the Choice:3

The elements in STACK

98
24
12
Press Next Choice
Enter the Choice:2

    The popped elements is 98
Enter the Choice:3

The elements in STACK

24
12
Press Next Choice
Enter the Choice:4

    EXIT POINT
```

---

Week 8

Write C programs that implement Queue (its operations) using

i) Arrays

ii) Pointers

```
/*
 * C Program to Implement a Queue using an Array
 */
#include <stdio.h>

#define MAX 50

void    insert();
void   delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
main()
```

```
{
   int choice;
   while (1)
   {
      printf("1.Insert element to queue \n");
      printf("2.Delete element from queue \n");
      printf("3.Display all elements of queue \n");
      printf("4.Quit \n");
      printf("Enter your choice : ");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
         insert();
         break;
         case 2:
         delete();
         break;
         case 3:
         display();
         break;
         case 4:
         exit(1);
         default:
         printf("Wrong choice \n");
      } /* End of switch */
   } /* End of while */
} /* End of main() */

void insert()
{
   int add_item;
   if (rear == MAX - 1)
   printf("Queue Overflow \n");
   else
   {
      if (front == - 1)
      /*If queue is initially empty */
      front = 0;
      printf("Inset the element in queue : ");
      scanf("%d", &add_item);
      rear = rear + 1;
      queue_array[rear] = add_item;
   }
```

```
} /* End of insert() */

void delete()
{
   if (front == - 1 || front > rear)
   {
      printf("Queue Underflow \n");
      return ;
   }
   else
   {
      printf("Element deleted from queue is : %d\n", queue_array[front]);
      front = front + 1;
   }
} /* End of delete() */

void display()
{
   int i;
   if (front == - 1)
      printf("Queue is empty \n");
   else
   {
      printf("Queue is : \n");
      for (i = front; i <= rear; i++)
         printf("%d ", queue_array[i]);
      printf("\n");
   }
} /* End of display() */
```

---

**/* Write C programs that implement Queue (its operations) using ii) Pointers */**

```
#define true 1
#define false 0

#include<stdio.h>
#include<conio.h>
#include<process.h>

struct q_point
{
int ele;
```

```
struct q_point* n;
};

struct q_point *f_ptr = NULL;

int e_que(void);
void  add_ele(int);
int rem_ele(void);
void show_ele();

/*main function*/
void main()
{
int ele,choice,j;
while(1)
{
clrscr();
printf("\n\n****IMPLEMENTATION OF QUEUE USING POINTERS****\n");
printf("==============================================");
printf("\n\t\t MENU\n");
printf("==============================================");
printf("\n\t[1] To insert an element");
printf("\n\t[2] To remove an element");
printf("\n\t[3] To display all the elements");
printf("\n\t[4] Exit");
printf("\n\n\tEnter your choice:");
scanf("%d", &choice);

switch(choice)
{
case 1:
{
printf("\n\tElement to be inserted:");
scanf("%d",&ele);
add_ele(ele);
getch();
break;
}
```



```
case 2:
{
```

```
if(!e_que())
{
j=rem_ele();
printf("\n\t%d is removed from the queue",j);
getch();
}
else
{
printf("\n\tQueue is Empty.");
getch();
}
break;
}

case 3:
show_ele();
getch();
break;

case 4:
exit(1);
break;

default:
printf("\n\tInvalid choice.");
getch();
break;
}

}
}

/* Function to check if the queue is empty*/
int e_que(void)
{
if(f_ptr==NULL)
return true;
return false;
}

/* Function to add an element to the queue*/
void add_ele(int ele)
{
struct q_point *queue = (struct q_point*)malloc(sizeof(struct q_point));
```

```
queue->ele = ele;
queue->n = NULL;
if(f_ptr==NULL)
f_ptr = queue;
else
{
struct q_point* ptr;
ptr = f_ptr;
for(ptr=f_ptr ;ptr->n!=NULL; ptr=ptr->n);
ptr->n = queue;
}
}

/* Function to remove an element from the queue*/
int rem_ele()
{
struct q_point* queue=NULL;
if(e_que()==false)
{
int j = f_ptr->ele;
queue=f_ptr;
f_ptr = f_ptr->n;
free (queue);
return j;
}
else
{
printf("\n\tQueue is empty.");
return -9999;
}
}

/* Function to display the queue*/
void show_ele()
{
struct q_point *ptr=NULL;
ptr=f_ptr;
if(e_que())
{
printf("\n\tQUEUE is Empty.");
return;
}
else
{
```

```
printf("\n\tElements present in Queue are:\n\t");
while(ptr!=NULL)
{
printf("%d\t",ptr->ele);
ptr=ptr->n;
}
}
}
```

Week 9

Write a C program that uses Stack operations to perform the following:

i) Converting infix expression into postfix expression

ii) Evaluating the postfix expression

### Infix to Postfix Conversion using the Stack Operations

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
char pop();
void push(char);
int pri(char);
char st[10];
int top=-1;
void main()
{
char s[25];
int i;
printf("Enter Infix Expression: ");
gets(s);
push('(');
strcat(s,")");
for(i=0;i<s[i];i++)
{
if(isalnum(s[i]))
printf("%c",s[i]);
else if(s[i]=='(')
push(s[i]);
else if(s[i]==')')
{
 while(st[top]!='(')
```

```
printf("%c",pop());
pop();
}
else
{
while(pri(s[i])<=pri(st[top]))
printf("%c",pop());
push(s[i]);
}
}
getch();
}
void push(char e)
{
st[++top]=e;
}
char pop()
{
char ch;
ch=st[top];
top--;
st[top+1]='\0';
return(ch);
}
int pri(char p)
{
switch(p)
{
case '(':return 0;
case '+':
case '-':return 1;
case '/':
case '%':
case '*':return 2;
}
}
```

Output

```
Enter Infix Expression: A+B
AB+

Enter Infix Expression: A+B*C-D
ABC*+D-
```

### Evaluation of Postfix Expressions Using Stack

```
/* This program is for evaluation of postfix expression
* This program assume that there are only four operators
* (*, /, +, -) in an expression and operand is single digit only
* Further this program does not do any error handling e.g.
* it does not check that entered postfix expression is valid
* or not.
* */

#include <stdio.h>
#include <ctype.h>

#define MAXSTACK 100 /* for max size of stack */
#define POSTFIXSIZE 100 /* define max number of charcters in postfix expression */

/* declare stack and its top pointer to be used during postfix expression
evaluation*/
int stack[MAXSTACK];
int top = -1; /* because array index in C begins at 0 */
/* can be do this initialization somewhere else */

/* define push operation */
void push(int item)
{

   if (top >= MAXSTACK - 1) {
      printf("stack over flow");
      return;
   }
   else {
      top = top + 1;
      stack[top] = item;
   }
}

/* define pop operation */
int pop()
{
   int item;
   if (top < 0) {
      printf("stack under flow");
   }
```

```
    else {
        item = stack[top];
        top = top - 1;
        return item;
    }
}

/* define function that is used to input postfix expression and to evaluate it */
void EvalPostfix(char postfix[])
{

    int i;
    char ch;
    int val;
    int A, B;

    /* evaluate postfix expression */
    for (i = 0; postfix[i] != ')'; i++) {
        ch = postfix[i];
        if (isdigit(ch)) {
            /* we saw an operand,push the digit onto stack
ch - '0' is used for getting digit rather than ASCII code of digit */
            push(ch - '0');
        }
        else if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
            /* we saw an operator
* pop top element A and next-to-top elemnet B
* from stack and compute B operator A
*/
            A = pop();
            B = pop();

            switch (ch) /* ch is an operator */
            {
            case '*':
                val = B * A;
                break;

            case '/':
                val = B / A;
                break;

            case '+':
                val = B + A;
```

```
            break;

        case '-':
            val = B - A;
            break;
        }

        /* push the value obtained above onto the stack */
        push(val);
    }
  }
  printf(" \n Result of expression evaluation : %d \n", pop());
}

int main()
{

  int i;

  /* declare character array to store postfix expression */
  char postfix[POSTFIXSIZE];
  printf("ASSUMPTION: There are only four operators(*, /, +, -) in an expression and operand is
single digit only.\n");
  printf(" \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");

  /* take input of postfix expression from user */

  for (i = 0; i <= POSTFIXSIZE - 1; i++) {
    scanf("%c", &postfix[i]);

    if (postfix[i] == ')') /* is there any way to eliminate this if */
    {
      break;
    } /* and break statement */
  }

  /* call function to evaluate postfix expression */

  EvalPostfix(postfix);

  return 0;
}
```

Output

First Run:
Enter postfix expression, press right parenthesis ')' for end expression : 456*+)
Result of expression evaluation : 34

Second Run:
Enter postfix expression, press right parenthesis ')' for end expression: 12345*+*+)
Result of expression evaluation: 47

---

Week 10

Write a C program that uses functions to perform the following operations on singly linked list.
i) Creation ii) Insertion iii) Deletion iv) Traversal

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int  count=0;
struct node
{
int data;
struct node *next;
}*head,*newn,*trav;
// -------------------------------------------------------
void create_list()
{
int value;
struct node *temp;
temp=head;
newn=(struct node *)malloc(sizeof (struct node));
printf("\nenter the value to be inserted");
scanf("%d",&value);
newn->data=value;
if(head==NULL)
{
head=newn;
head->next=NULL;
count++;
}
else
{
while(temp->next!=NULL)
{
```

```
temp=temp->next;
}
temp->next=newn;
newn->next=NULL;
count++;
}
}
// ----------------------------------------------------
void insert_at_begning(int value)
{
newn=(struct node *)malloc(sizeof (struct node));
newn->data=value;
if(head==NULL)
{
head=newn;
head->next=NULL;
count++;
}
else
{
newn->next=head;
head=newn;
count++;
}
}
// ---------------------------------------------------------
void insert_at_end(int value)
{
struct node *temp;
temp=head;
newn=(struct node *)malloc(sizeof (struct node));
newn->data=value;
if(head==NULL)
{
head=newn;
head->next=NULL;
count++;
}
else
{
while(temp->next!=NULL)
{
temp=temp->next;
}
```

```
temp->next=newn;
newn->next=NULL;
count++;
}
}
// ----------------------------------------------------
int insert_at_middle()
{
if(count>=2)
{
struct node *var1,*temp;
int loc,value;
printf("\n after which value you want to insert : ");
scanf("%d",&loc);
printf("\nenter the value to be inserted");
scanf("%d",&value);
newn=(struct node *)malloc(sizeof (struct node));
newn->data=value;
temp=head;

/* if(head==NULL)
{
head=newn;
head->next=NULL;
count++;
return 0;
}
else
{*/
while(temp->data!=loc)
{
temp=temp->next;
if(temp==NULL)
{
printf("\nSORRY...there is no %d element",loc);
return 0;
}
}
//var1=temp->next;
newn->next=temp->next;//var1;
temp->next=newn;

count++;
//}
```

```
}
else
{
printf("\nthe no of nodes must be >=2");
}
}
// ------------------------------------------------------------
int delete_from_middle()
{
if(count==0)
printf("\n List is Empty!!!! you can't delete elements\n");
else if(count>2)
{
struct node *temp,*var;
int value;
temp=head;
printf("\nenter the data that you want to delete from the list shown above");
scanf("%d",&value);
while(temp->data!=value)
{
var=temp;
temp=temp->next;
if(temp==NULL)
{
printf("\nSORRY...there is no %d element",value);
return 0;
}
}
if(temp==head)
{
head=temp->next;

}
else{
var->next=temp->next;
temp->next=NULL;

}
count--;
if(temp==NULL)
printf("Element is not avilable in the list \n**enter only middle elements..**");
else
printf("\ndata deleted from list is %d",value);
free(temp);
```

```
}
else
{
printf("\nthere no middle elemts..only %d elemts is avilable\n",count);
}
}
// ----------------------------------------------------------------
int delete_from_front()
{
struct node *temp;
temp=head;
if(head==NULL)
{
printf("\nno elements for deletion in the list\n");
return 0;
}
else
{
printf("\ndeleted element is :%d",head->data);
if(temp->next==NULL)
{
head=NULL;
}
else{
head=temp->next;
temp->next=NULL;

}
count--;
free(temp);

}
}
// -------------------------------------------------------
int delete_from_end()
{
struct node *temp,*var;
temp=head;
if(head==NULL)
{
printf("\nno elemts in the list");
return 0;
}
else{
```

```
if(temp->next==NULL )
{
head=NULL;//temp->next;
}
else{
while(temp->next != NULL)
{
var=temp;
temp=temp->next;
}
var->next=NULL;
}
printf("\ndata deleted from list is %d",temp->data);
free(temp);
count--;
}
return 0;
}
// -------------------------------------------------------
int display()
{
trav=head;
if(trav==NULL)
{
printf("\nList is Empty\n");
return 0;
}
else
{
printf("\n\nElements in the Single Linked List is %d:\n",count);
while(trav!=NULL)
{
printf(" -> %d ",trav->data);
trav=trav->next;
}
printf("\n");
}
}
// ---------------------------------------------------------------
int main()
{
int ch=0;
char ch1;
```

```
head=NULL;
while(1)
{
printf("\n1.create linked list");
printf("\n2.insertion at begning of linked list");
printf("\n3.insertion at the end of linked list");
printf("\n4.insertion at the middle where you want");
printf("\n5.deletion from the front of linked list");
printf("\n6.deletion from the end of linked list ");
printf("\n7.deletion of the middle data that you want");
printf("\n8.display the linked list");
printf("\n9.exit\n");
printf("\nenter the choice of operation to perform on linked list");
scanf("%d",&ch);
switch(ch)
{
case 1:
{
do{
create_list();
display();
printf("do you want to create list ,y / n");
getchar();
scanf("%c",&ch1);
}while(ch1=='y');

break;
}
case 2:
{
int value;
printf("\nenter the value to be inserted");
scanf("%d",&value);
insert_at_begning(value);
display();
break;
}
case 3:
{
int value;
printf("\nenter value to be inserted");
scanf("%d",&value);
insert_at_end(value);
display();
```

```
break;
}
case 4:
{
insert_at_middle();
display();
break;
}
case 5:
{
delete_from_front();
display();
}break;
case 6:
{
delete_from_end();
display();
break;
}
case 7:
{
display();
delete_from_middle();
display();
break;
}
case 8:
{
display();
break;
}
case 9:
{
exit(1);
}
default:printf("\n****Please enter correct choice****\n");
}
}
getch();
}
```

**Output:**

```
Activities  Terminal   Mon 9:06 PM   student
student@localhost:~/187Z1A0515
File  Edit  View  Search  Terminal  Help
[student@localhost 187Z1A0515]$ gcc sll.c
[student@localhost 187Z1A0515]$ ./a.out

operations on single linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        1
insert any value into the node
10
the data available is
10->
operations on single linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        1
insert any value into the node
20
```

```
Activities  Terminal   Mon 9:08 PM   student
student@localhost:~/187Z1A0515
File  Edit  View  Search  Terminal  Help
the data available is
10->20->
operations on single linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        1
insert any value into the node
30
the data available is
10->20->30->
operations on single linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        2
insert any value into the node
50
the data available is
```

Week 11

Write a C program that uses functions to perform the following operations on Doubly linkedlist.

i) Creation ii) Insertion iii) Deletion iv) Traversal

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int  count=0;
struct node
{
int data;
struct node *next,*prev;
}*head,*last,*newn,*trav;
// ---------------------------------------------------
void create_list()
{
struct node *temp;
int value;
temp=last;
newn=(struct node *)malloc(sizeof (struct node));
printf("\n enter value");
scanf("%d",&value);
newn->data=value;
if(last==NULL)
{
head=last=newn;
head->prev=NULL;
head->next=NULL;
count++;
}
else
{
newn->next=NULL;
newn->prev=last;
last->next=newn;
last=newn;
count++;
}
}
// ---------------------------------------------------
void insert_at_begning(int value)
{
newn=(struct node *)malloc(sizeof (struct node));
newn->data=value;
if(head==NULL)
{
head=last=newn;
```

```
head->prev=NULL;
head->next=NULL;
count++;
}
else
{
newn->next=head;
head->prev=newn;
newn->prev=NULL;
head=newn;
count++;
}
}
// -------------------------------------------------------
void insert_at_end(int value)
{
struct node *temp;
temp=last;
newn=(struct node *)malloc(sizeof (struct node));
newn->data=value;
if(last==NULL)
{
head=last=newn;
head->prev=NULL;
head->next=NULL;
count++;
}
else
{
newn->next=NULL;
newn->prev=last;
last->next=newn;
last=newn;
count++;
}
}
// -----------------------------------------------------
int insert_at_middle()
{
if(count>=2)
{
struct node *var2,*temp;
int loc,value;
printf("\nselect location where you want to insert the data");
```

```
scanf("%d",&loc);
printf("\nenter which value do u want to inserted");
scanf("%d",&value);
newn=(struct node *)malloc(sizeof (struct node));
newn->data=value;
temp=head;
while(temp->data!=loc)
{
temp=temp->next;
if(temp==NULL)
{
printf("\nSORRY...there is no %d element",loc);
return 0;
}
}
if(temp->next==NULL)
{
printf("\n%d is last node..please enter midddle node values ",loc) ;
return;
}
var2=temp->next;
temp->next=newn;
newn->next=var2;
newn->prev=temp;
var2->prev=newn;
count++;
}
else
{
printf("\nthe no of nodes must be >=2");
}
}
// ------------------------------------------------------------
int delete_from_middle()
{
if(count>2)
{
struct node *temp,*var;
int value;
temp=head;
printf("\nenter the data that you want to delete from the list shown above");
scanf("%d",&value);
while(temp!=NULL)
{
```

```
if(temp->data == value)
{
if(temp->next==NULL)//if(temp==head)
{
printf("\n\n sorry %d is last node ..please enter middle nodes only",value);
return 0;
}
else
{
if(temp==head)
{
printf("\n\n %d is first node..plz enter middle nodes",value);
return;

}
var->next=temp->next;
temp->next->prev=var;
free(temp);
count--;
return 0;
}
}
else
{
var=temp;
temp=temp->next;
}
}
if(temp==NULL)
printf("\n%d is not avilable.. enter only middle elements..",value);
else
printf("\ndata deleted from list is %d",value);
}
else
{
printf("\nthere no middle elemts..only %d elemts is avilable",count);
}
}
// ----------------------------------------------------------------
int delete_from_front()
{
struct node *temp;
if(head==NULL)
{
```

```
printf("no elements for deletion in the list");
return 0;
}
else if(head->next==NULL)
{
printf("deleted element is :%d",head->data);
head=last=NULL;

}
else
{
temp=head->next;
printf("deleted element is :%d",head->data);
head->next=NULL;
temp->prev=NULL;
head=temp;
count--;
return 0;
}
}
// -------------------------------------------------------
int delete_from_end()
{
struct node *temp,*var;
temp=last;
if(last==NULL)
{
printf("no elemts in the list");
return 0;
}
else if(last->prev==NULL)
{
printf("data deleted from list is %d",last->data);
head=last=NULL;
//free(last);
count--;
return 0;
}
else{
printf("data deleted from list is %d",last->data);
var=last->prev;
last->prev->next=NULL;
last=var;
count--;
```

```
return 0;
}
}
// ----------------------------------------------------
int display()
{
trav=last;//head;
if(trav==NULL)
{
printf("\nList is Empty");
return 0;
}
else
{
printf("\n\nElements in the List is %d:\n",count);
while(trav!=NULL)
{
printf("%d<--> ",trav->data);
trav=trav->prev;//next;
}
printf("\n");
}
}
// ------------------------------------------------------------
int main()
{
int ch=0;
char ch1;
// clrscr();
head=NULL;
last=NULL;
while(1)
{
Printf("\n Double Linked List Operations")
printf("\n1.Create Double Linked List");
printf("\n2.insertion at begning of linked list");
printf("\n3.insertion at the end of linked list");
printf("\n4.insertion at the middle where you want");
printf("\n5.deletion from the front of linked list");
printf("\n6.deletion from the end of linked list ");
printf("\n7.deletion of the middle data that you want");
printf("\n8.display");
printf("\n9.exit\n");
printf("\nenter the choice of operation to perform on linked list");
```

```
scanf("%d",&ch);
switch(ch)
{
case 1:
{
do{
create_list();
display();
printf("do you want to create list ,y / n");
getchar();
scanf("%c",&ch1);
}while(ch1=='y');
break;
}
case 2:
{
int value;
printf("\nenter the value to be inserted");
scanf("%d",&value);
insert_at_begning(value);
display();
break;
}
case 3:
{
int value;
printf("\nenter value to be inserted");
scanf("%d",&value);
insert_at_end(value);
display();
break;
}
case 4:
{
insert_at_middle();
display();
break;
}
case 5:
{
delete_from_front();
display();
}break;
case 6:
```

```
{
delete_from_end();
display();
break;
}
case 7:
{
display();
delete_from_middle();
display();
break;
}
case 8:display();break;
case 9:
{
exit(0);
}
}
}
getch();
}
```

**Output:**

```
Activities  Terminal  Mon 9:11 PM  student
student@localhost:~/187Z1A0515
File Edit View Search Terminal Help
[student@localhost 187Z1A0515]$ gcc dll.c
[student@localhost 187Z1A0515]$ ./a.out

operations on double linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        1
enter the value10
10
operations on double linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        1
enter the value20
10<->20
operations on double linked list
1.creat
```

```
Activities  Terminal  Mon 9:12 PM  student
student@localhost:~/187Z1A0515
File Edit View Search Terminal Help
10<->20
operations on double linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        1
enter the value30
10<->20<->30
operations on double linked list
1.creat
2.insert at front
3.insert at middle
4.insert at end
5.display
6.delete at front
7.delete at middle
8.delete at end
9.exit
Enter ur choice:        2
insert any value into the node
50
50<->10<->20<->30
operations on double linked list
1.creat
2.insert at front
```

Week 12

Write a C program that uses functions to perform the following operations on circular linkedlist.

i) Creation ii) Insertion iii) Deletion iv) Traversal

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
struct node *head;
void beginsert ();
void lastinsert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main ()
{
int choice =0;
while(choice != 7)
{
printf("\n*********Main Menu*********\n");
printf("\nChoose one option from the following list ...\n");
printf("\n==============================================\n");
printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from Beginning\n4.Delete from
last\n5.Search for an element\n6.Show\n7.Exit\n");
printf("\nEnter your choice?\n");
scanf("\n%d",&choice);
switch(choice)
{
case 1:
beginsert();
break;
case 2:
lastinsert();
break;
case 3:
begin_delete();
break;
```

```
case 4:
last_delete();
break;
case 5:
search();
break;
case 6:
display();
break;
case 7:
exit(0);
break;
default:
printf("Please enter valid choice..");
}
}
}
void beginsert()
{
struct node *ptr,*temp;
int item;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW");
}
else
{
printf("\nEnter the node data?");
scanf("%d",&item);
ptr -> data = item;
if(head == NULL)
{
head = ptr;
ptr -> next = head;
}
else
{
temp = head;
while(temp->next != head)
temp = temp->next;
ptr->next = head;
temp -> next = ptr;
head = ptr;
```

```
}
printf("\nnode inserted\n");
}
}
void lastinsert()
{
struct node *ptr,*temp;
int item;
ptr = (struct node *)malloc(sizeof(struct node));
if(ptr == NULL)
{
printf("\nOVERFLOW\n");
}
else
{
printf("\nEnter Data?");
scanf("%d",&item);
ptr->data = item;
if(head == NULL)
{
head = ptr;
ptr -> next = head;
}
else
{
temp = head;
while(temp -> next != head)
{
temp = temp -> next;
}
temp -> next = ptr;
ptr -> next = head;
}

printf("\nnode inserted\n");
}
}

void begin_delete()
{
struct node *ptr;
if(head == NULL)
{
printf("\nUNDERFLOW");
```

```
}
else if(head->next == head)
{
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else
{  ptr = head;
while(ptr -> next != head)
ptr = ptr -> next;
ptr->next = head->next;
free(head);
head = ptr->next;
printf("\nnode deleted\n");
}
}
void last_delete()
{
struct node *ptr, *preptr;
if(head==NULL)
{
printf("\nUNDERFLOW");
}
else if (head ->next == head)
{
head = NULL;
free(head);
printf("\nnode deleted\n");
}
else
{
ptr = head;
while(ptr ->next != head)
{
preptr=ptr;
ptr = ptr->next;
}
preptr->next = ptr -> next;
free(ptr);
printf("\nnode deleted\n");
}
}
void search()
```

```
{
struct node *ptr;
int item,i=0,flag=1;
ptr = head;
if(ptr == NULL)
{
printf("\nEmpty List\n");
}
else
{
printf("\nEnter item which you want to search?\n");
scanf("%d",&item);
if(head ->data == item)
{
printf("item found at location %d",i+1);
flag=0;
}
else
{
while (ptr->next != head)
{
if(ptr->data == item)
{
printf("item found at location %d ",i+1);
flag=0;
break;
}
else
{
flag=1;
}
i++;
ptr = ptr -> next;
}
}
if(flag != 0)
{
printf("Item not found\n");
} } }

void display()
{
struct node *ptr;
ptr=head;
```

```
if(head == NULL)
{
printf("\nnothing to print");
}
else
{
printf("\n printing values ... \n");
while(ptr -> next != head)
{
printf("%d\n", ptr -> data);
ptr = ptr -> next;
}
printf("%d\n", ptr -> data);
}
}
```

**Output:**

```
Activities  Terminal                Mon 9:33 PM                    student
                    student@localhost:~/187Z1A0515
File  Edit  View  Search  Terminal  Help
[student@localhost 187Z1A0515]$ gcc CLL.c
[student@localhost 187Z1A0515]$ ./a.out

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
1

Enter the node data?10

node inserted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in begining
```

```
Activities  Terminal                Mon 9:34 PM                    student
                    student@localhost:~/187Z1A0515
File  Edit  View  Search  Terminal  Help

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
1

Enter the node data?20

node inserted

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
```

```
// C program to evaluate value of a postfix expression
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

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

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

   if (!stack) return NULL;

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

   if (!stack->array) return NULL;

   return stack;
}

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

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

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

```
    return '$';
}

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


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

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

    // Scan all characters one by one
    for (i = 0; exp[i]; ++i)
    {
        // If the scanned character is an operand (number here),
        // push it to the stack.
        if (isdigit(exp[i]))
            push(stack, exp[i] - '0');

        // If the scanned character is an operator, pop two
        // elements from stack apply the operator
        else
        {
            int val1 = pop(stack);
            int val2 = pop(stack);
            switch (exp[i])
            {
            case '+': push(stack, val2 + val1); break;
            case '-': push(stack, val2 - val1); break;
            case '*': push(stack, val2 * val1); break;
            case '/': push(stack, val2/val1); break;
            }
        }
    }
    return pop(stack);
}
```

```
// Driver program to test above functions
int main()
{
   char exp[] = "231*+9-";
   printf ("postfix evaluation: %d", evaluatePostfix(exp));
   return 0;
}
```

**Output**

postfix evaluation: -4

---

Week 14

Write C programs that use both recursive and non-recursive functions to perform the following searching operations for a key value in a given list of integers:

i) Linear search

ii) Binary search

```
// Write C programs that use both recursive and non recursive functions
// to perform the following searching operation for a Key value in a given list of integers :
// i) Linear search

#include <stdio.h>
#include <conio.h>
#define MAX_LEN 10

void l_search_recursive(int l[],int num,int ele);
void l_search_nonrecursive(int l[],int num,int ele);
void read_list(int l[],int num);
void print_list(int l[],int num);

int main()
{
   int l[MAX_LEN], num, ele;
   int ch;

   printf("=====================================================");
   printf("\n\t\t\tMENU");
   printf("\n=====================================================");
   printf("\n[1] Linary Search using Recursion method");
   printf("\n[2] Linary Search using Non-Recursion method");
   printf("\n\nEnter your Choice:");
   scanf("%d",&ch);
```

```
    if(ch<=2 & ch>0)
    {
        printf("Enter the number of elements :");
        scanf("%d",&num);
        read_list(l,num);
        printf("\nElements present in the list are:\n\n");
        print_list(l,num);
        printf("\n\nElement you want to search:\n\n");
        scanf("%d",&ele);

        switch(ch)
        {
        case 1:
            printf("\n**Recursion method**\n");
            l_search_recursive(l,num,ele);
            getch();
            break;

        case 2:
            printf("\n**Non-Recursion method**\n");
            l_search_nonrecursive(l,num,ele);
            getch();
            break;
        }
    }
    getch();
}

//end main

//Non-Recursive method

void l_search_nonrecursive(int l[],int num,int ele)
{
    int j, f=0;
    for(j=0; j<num; j++)
        if( l[j] == ele)
        {
            printf("\nThe element %d is present at position %d in list\n",ele,j);
            f=1;
            break;
        }
    if(f==0)
```

```
        printf("\nThe element is %d is not present in the list\n",ele);
}

//Recursive method

void l_search_recursive(int l[],int num,int ele)
{
    int f = 0;

    if( l[num] == ele)
    {
        printf("\nThe element %d is present at position %d in list\n",ele,num);
        f=1;
    }
    else
    {
        if((num==0) && (f==0))
        {
            printf("The element %d is not found.",ele);
        }
        else
        {
            l_search_nonrecursive(l,num-1,ele);
        }
    }
    getch();
}

void read_list(int l[],int num)
{
    int j;
    printf("\nEnter the elements:\n");
    for(j=0; j<num; j++)
        scanf("%d",&l[j]);
}

void print_list(int l[],int num)
{
    int j;
    for(j=0; j<num; j++)
        printf("%d\t",l[j]);
}
```

**Write a C program that uses non recursive function to search for a Key value in a given sorted list of integers using Binary search.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[20], i, n, key, low, high, mid;
 clrscr();
 printf("Enter the array elements in ascending order");
 for(i = 0; i < n; i++)
 {
  scanf("%d", &a[i]);
 }
 printf("Enter the key element\n");
 scanf("%d", &key);
 low = 0;
 high = n - 1;
 while(high >= low)
 {
  mid = (low + high) / 2;
  if(key == a[mid])
  break;
  else
  {
   if(key > a[mid])
   low = mid + 1;
   else
    high = mid - 1;
  }
 }
 if(key == a[mid])
  printf("The key element is found at location %d", mid + 1);
 else
  printf("the key element is not found");
 getch();
}
```

**Input & Output:**

```
Enter the size of the array 7
Enter the array elements in ascending order 23 45 68 90 100 789 890
Enter the key element 789
The key Element is found at location 6
```

Week 15

Write a C program that implements the following sorting methods to sort a given list of integers in

ascending order

i) Bubble sort

ii) Selection sort

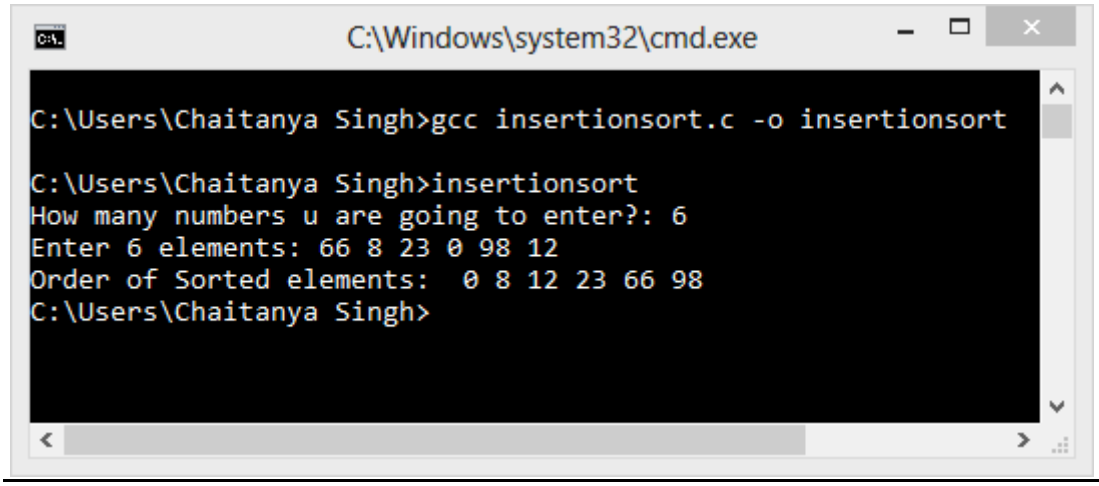
iii) Insertion sort

**C Program – Insertion Sort implementation**

```
#include<stdio.h>
int main(){

  /* Here i & j for loop counters, temp for swapping,
   * count for total number of elements, number[] to
   * store the input numbers in array. You can increase
   * or decrease the size of number array as per requirement
   */
  int i, j, count, temp, number[25];

  printf("How many numbers u are going to enter?: ");
  scanf("%d",&count);

  printf("Enter %d elements: ", count);
  // This loop would store the input numbers in array
  for(i=0;i<count;i++)
    scanf("%d",&number[i]);

  // Implementation of insertion sort algorithm
  for(i=1;i<count;i++){
    temp=number[i];
    j=i-1;
    while((temp<number[j])&&(j>=0)){
      number[j+1]=number[j];
      j=j-1;
    }
    number[j+1]=temp;
  }

  printf("Order of Sorted elements: ");
  for(i=0;i<count;i++)
```

```
    printf(" %d",number[i]);

  return 0;
}
```

**Output:**

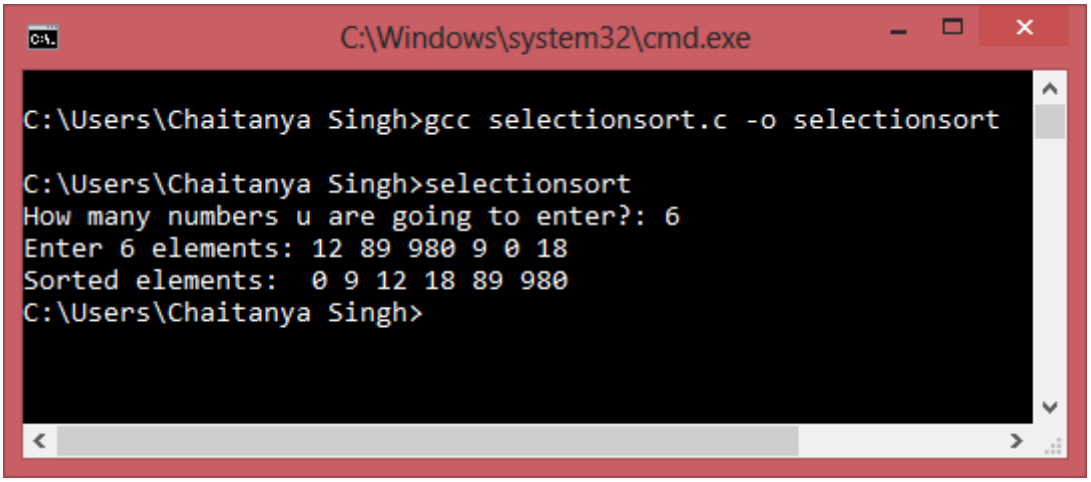



**C Program – Selection sort**

```
#include<stdio.h>
int main(){
  /* Here i & j for loop counters, temp for swapping,
   * count for total number of elements, number[] to
   * store the input numbers in array. You can increase
   * or decrease the size of number array as per requirement
   */
  int i, j, count, temp, number[25];

  printf("How many numbers u are going to enter?: ");
  scanf("%d",&count);

  printf("Enter %d elements: ", count);
  // Loop to get the elements stored in array
  for(i=0;i<count;i++)
    scanf("%d",&number[i]);

  // Logic of selection sort algorithm
  for(i=0;i<count;i++){
    for(j=i+1;j<count;j++){
      if(number[i]>number[j]){
        temp=number[i];
        number[i]=number[j];
```

```
            number[j]=temp;
        }
      }
    }

    printf("Sorted elements: ");
    for(i=0;i<count;i++)
      printf(" %d",number[i]);

    return 0;
}
```

**Output:**




---

---

**Implementing bubble sort algorithm in a C program**

```
/* Implementing Bubble sort in a C Program
 * Written by: Chaitanya.
 */
#include<stdio.h>

int main(){

    int count, temp, i, j, number[30];

    printf("How many numbers are u going to enter?: ");
    scanf("%d",&count);

    printf("Enter %d numbers: ",count);
```

```
  for(i=0;i<count;i++)
  scanf("%d",&number[i]);

  /* This is the main logic of bubble sort algorithm
   */
  for(i=count-2;i>=0;i--){
    for(j=0;j<=i;j++){
    if(number[j]>number[j+1]){
        temp=number[j];
        number[j]=number[j+1];
        number[j+1]=temp;
      }
    }
  }

  printf("Sorted elements: ");
  for(i=0;i<count;i++)
    printf(" %d",number[i]);

  return 0;
}
```

**Output:**