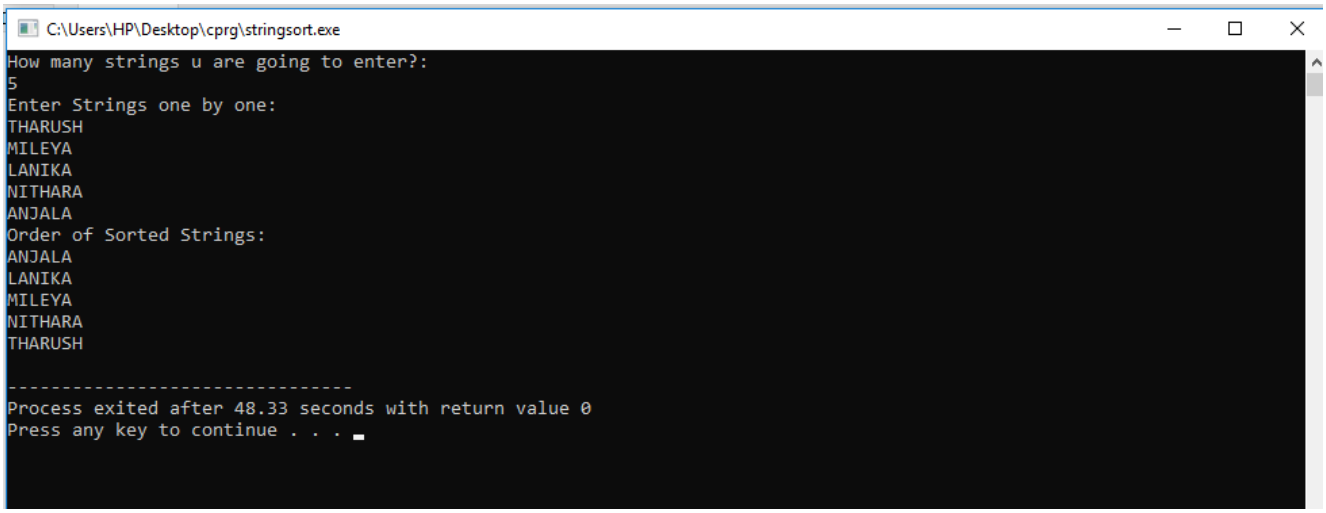
DATA STRUCTURE PROGRAMMING

COMPUTER APPLICATION - IV SEMESTER

1. SORT A GIVEN LIST OF STRINGS

```
#include<stdio.h>
#include<string.h>
int main()
{
  int i,j,count;
  char str[25][25],temp[25];
  puts("How many strings u are going to enter?: ");
  scanf("%d",&count);
  puts("Enter Strings one by one: ");
  for(i=0;i<=count;i++)
    gets(str[i]);
  for(i=0;i<=count;i++)
    for(j=i+1;j<=count;j++)
{
      if(strcmp(str[i],str[j])>0)
{
        strcpy(temp,str[i]);
        strcpy(str[i],str[j]);
        strcpy(str[j],temp);
      }
    }
  printf("Order of Sorted Strings:");
  for(i=0;i<=count;i++)
    puts(str[i]);

  return 0;
}
```

OUTPUT

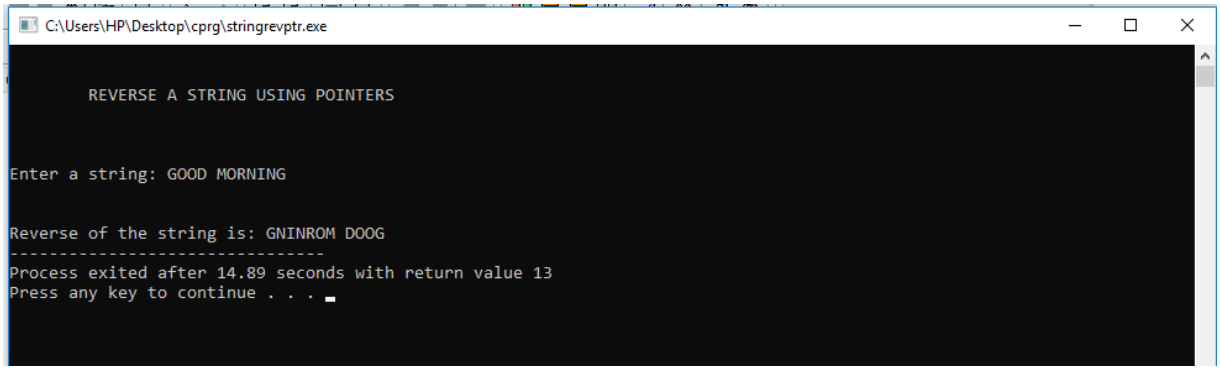


2. REVERSE A STRING USING POINTER

```
#include <stdio.h>
#include<string.h>
void main()
{
   printf("\n\n\tREVERSE A STRING USING POINTERS\n\n");
   char str[100];
   char rev[100];
   char *sptr = str; // sptr stores the base address of the str
   char *rptr = rev; // rptr stores the base address of the reverse
   int i = -1;
   printf("\n\nEnter a string: ");
   gets(str);
   // storing the ending address of str in sptr
   while(*sptr)
   {
      sptr++;
      i++; // i is the index of the end location
```

```
    }
    // storing the string str in rev in reverse order
    while(i >= 0)
    {
        /* First decrementing then using as it stores
            the location after the end location due to above while loop  */
        sptr--;
        *rptr = *sptr;  // storing the value in sptr in rptr
        rptr++; // pointing to next location
        i--;    // decrementing the index
    }
    /*
        String should always end with '\0' so explicitly
        putting it at the end of the string
    */
    *rptr = '\0';
    rptr = rev; // restoring the base address of the reverse string
    // storing the reverse string in the original string
    while(*rptr)
    {
        *sptr = *rptr;
        sptr++;
        rptr++;
    }
    // printing the reverse string
    printf("\n\nReverse of the string is: %s ", str);
      getch();
}
```

OUTPUT

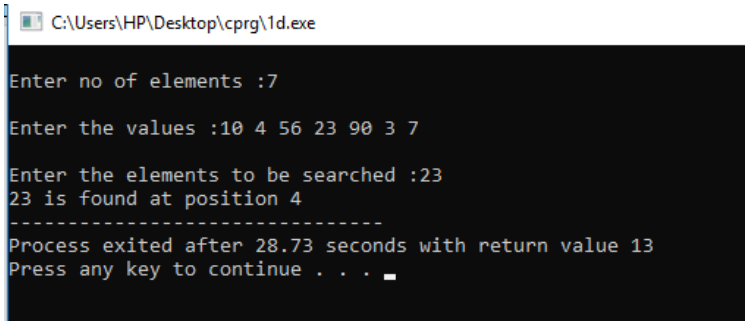


3. SEARCH AN ELEMENT IN A 1D ARRAY

```
#include<stdio.h>
void main()
{
  int a[30], ele, num, i,flag=0;
   printf("\nEnter no of elements :");
  scanf("%d", &num);
   printf("\nEnter the values :");
  for (i = 0; i < num; i++)
  {
    scanf("%d", &a[i]);
  }
   printf("\nEnter the elements to be searched :");
  scanf("%d", &ele);
  for(i=0;i<num;i++)
  {
        if(a[i]==ele)
        {
              flag=1;
              break;
```

```
            }
}
  if(flag==1)
  printf("%d is found at position %d",ele,i+1);
  else
   printf("%d is not found in the array ",ele);
getch();
  }
```

OUTPUT

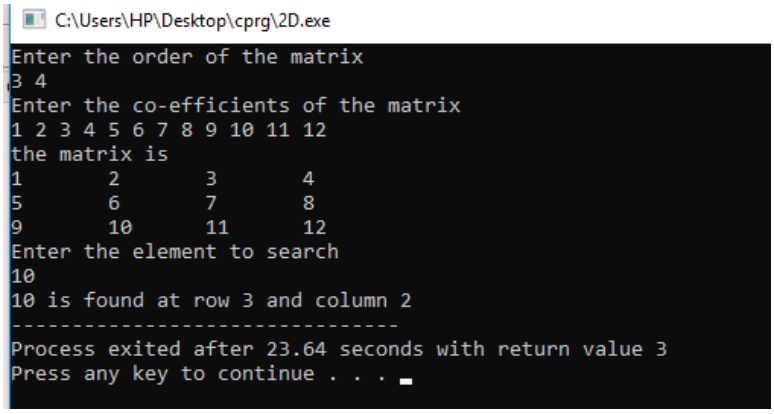


4. SEARCH AN ELEMENT IN AN 2D ARRAY

```
#include<stdio.h>
main()
  {
     int array[10][10];
     int i, j, r, c, sum = 0,flag=0,item;
      printf("Enter the order of the matrix\n");
     scanf("%d %d", &r, &c);
      printf("Enter the co-efficients of the matrix\n");
     for (i = 0; i < r; ++i)
     {
        for (j = 0; j < c; ++j)
        {
           scanf("%d", &array[i][j]);
        }
```

```
        }
 printf("the matrix is \n");
        for (i = 0; i < r; ++i)
        {
            for (j = 0; j < c; ++j)
            {
                printf("%d\t", array[i][j]);
            }
            printf("\n");
        }
 printf("Enter theELEMENT TO SEARCH\n");
 scanf("%d",&item);
        for (i = 0; i < r; ++i)
        {
            for (j = 0; j < c; ++j)
            {
                if(array[i][j] ==item)
                {
                        flag=1;
                 printf("%d is found at row %d and column %d",item,i,j);
                                        }
            }
}

 if(flag==0)
 printf("%d is  not found in the 2D array",item);
   }
```

OUTPUT

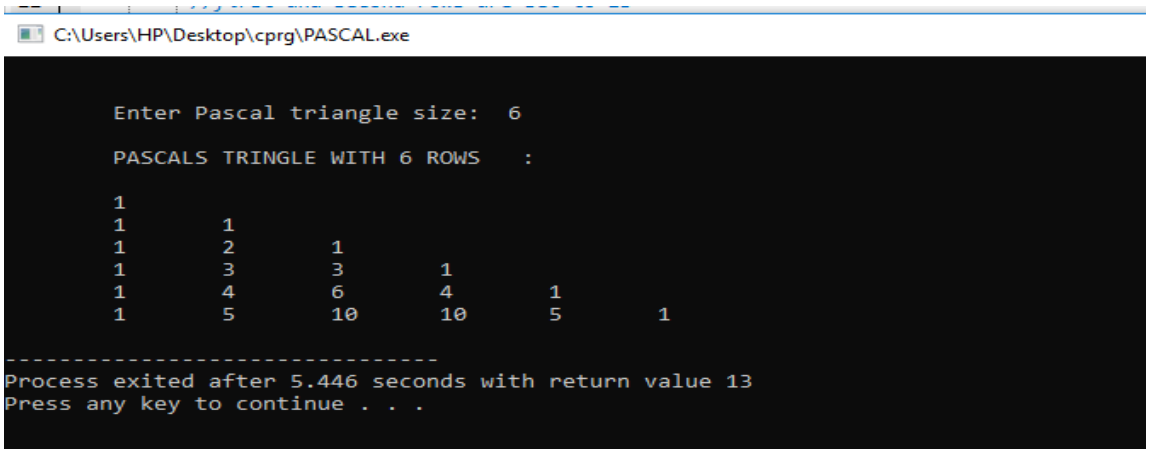


5. IMPLEMENT PASCALS TRIANGLE USING 2D ARRAY

```
#include<stdio.h>
void printPascalTr(int size);
void main()
   {
    int size;
    printf("\n\n\tEnter Pascal triangle size:  ");
    scanf("%d",&size);
    printPascalTr(size);
    getch();
 }
 void printPascalTr(int size)
{
       int PascalTr[size][size];
       int row,col;
       //assign zero to every array element
       for(row=0;row<size;row++)
       for(col=0;col<size;col++)
       PascalTr[row][col]=0;
       //first and second rows are set to 1s
       PascalTr[0][0]=1;
       PascalTr[1][0]=1;
       PascalTr[1][1]=1;
       for(row=2;row<size;row++)
       {
```

```
        PascalTr[row][0]=1;
        for(col=1;col<=row;col++)
       {
          PascalTr[row][col]=PascalTr[row-1][col-1]+PascalTr[row-1][col];
        }
      }
      printf("\n\tPASCALS TRINGLE WITH %d ROWS   :  \n\n",row);
      for(row=0;row<size;row++)
     {
       for(col=0;col<=row;col++)
      {
         printf("\t%d",PascalTr[row][col]);
       }
       printf("\n");
      }
    }
```

OUTPUT



6. MERGE 2 SORTED ARRAYS INTO 1 SORTED ARRAY

```
#include <stdio.h>
  int main()
  {
     int array1[50], array2[50], array3[100], m, n, i, j, k = 0;
    printf("\n Enter size of array Array 1: ");
```

```
scanf("%d", &m);
 printf("\n Enter sorted elements of array 1: \n");
for (i = 0; i < m; i++)
{
   scanf("%d", &array1[i]);
}

printf("\n Enter size of array 2: ");
scanf("%d", &n);
 printf("\n Enter sorted elements of array 2: \n");
for (i = 0; i < n; i++)
{
   scanf("%d", &array2[i]);
}
 i = 0;
j = 0;
 while (i < m && j < n)
{
   if (array1[i] < array2[j])
   {
      array3[k] = array1[i];
      i++;
   }
    else
   {
      array3[k] = array2[j];
      j++;
   }
   k++;
}
 if (i >= m)
{
```

```
        while (j < n)
        {
            array3[k] = array2[j];
            j++;
            k++;
        }
    }
  if (j >= n)
    {
        while (i < m)
        {
            array3[k] = array1[i];
            i++;
            k++;
        }
    }

    printf("\n After merging: \n");
    for (i = 0; i < m + n; i++)
    {
        printf("\n%d", array3[i]);
    }
   return 0;
  }
```

OUTPUT

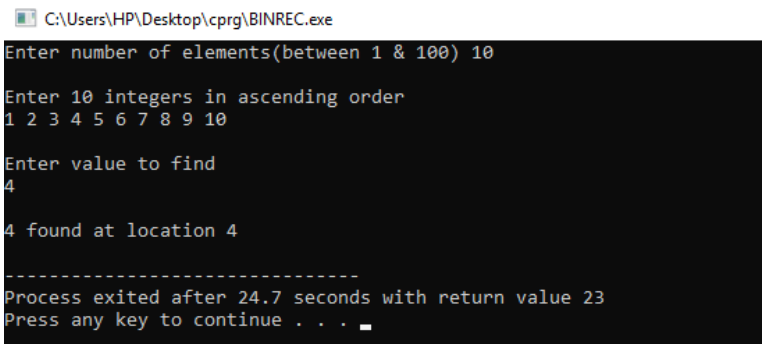
```
 Enter size of array Array 1: 4

 Enter sorted elements of array 1:
1 5 8 10

 Enter size of array 2: 5

 Enter sorted elements of array 2:
2 3 9 11 15

 After merging:

1
2
3
5
8
9
10
11
15
--------------------------------
Process exited after 193.8 seconds with return value 0
Press any key to continue . . .
```

7. SEARCH AN ELEMENT IN AN ARRAY USING RECURSIVE BINARY SEARCH

```
#include <stdio.h>
void binary_search(int [], int, int, int);
void main()
{
   int key, n, i;
   int array[100];
   printf("Enter number of elements(between 1 & 100) ");
   scanf("%d", &n);
   printf("\nEnter %d integers in ascending order\n", n);
   for(i = 0; i < n; i++)
   {
      scanf("%d",&array[i]);
   }
   printf("\n");
   printf("Enter value to find\n");
   scanf("%d", &key);
   binary_search(array, 0, n, key);
 }
 void binary_search(int array[], int low, int high, int key)
{
```

```
    int mid;
     if (low > high)
    {
        printf(" %d isn't present in the list or list is not in ascending order\n", key);
        return;
    }
    mid = (low + high) / 2;
    if (array[mid] == key)
    {
        printf("\n%d found at location %d\n", key, mid+1);
    }
    else if (array[mid] > key)
    {
        binary_search(array, low, mid - 1, key);
    }
    else if (array[mid] < key)
    {
        binary_search(array, mid + 1, high, key);
    }
}
```

OUTPUT



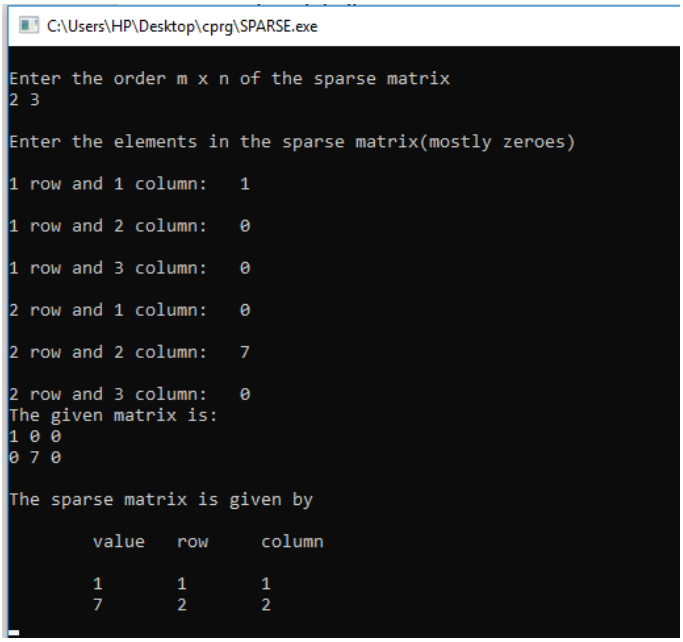
8. IMPLEMENT SPARSE MATRIX

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

```
int A[10][10],B[10][3],m,n,s=0,i,j;
printf("\nEnter the order m x n of the sparse matrix\n");
scanf("%d%d",&m,&n);
printf("\nEnter the elements in the sparse matrix(mostly zeroes)\n");
for(i=0;i<m;i++)
{
 for(j=0;j<n;j++)
 {
  printf("\n%d row and %d column:   ",i+1,j+1);
  scanf("%d",&A[i][j]);
 }
}
printf("The given matrix is:\n");
for(i=0;i<m;i++)
{
 for(j=0;j<n;j++)
 {
  printf("%d ",A[i][j]);
 }
 printf("\n");
}
for(i=0;i<m;i++)
{
 for(j=0;j<n;j++)
 {
  if(A[i][j]!=0)
  {
   B[s][0]=A[i][j];
   B[s][1]=i+1;
   B[s][2]=j+1;
   s++;
  }
```

```
 }
 }
 printf("\nThe sparse matrix is given by\n");
 printf("\n");
 printf("\tvalue\trow\tcolumn\n\n");
 for(i=0;i<s;i++)
 {
  for(j=0;j<3;j++)
  {
   printf("\t%d ",B[i][j]);
  }
  printf("\n");
 }
 getch();
}
```

OUTPUT

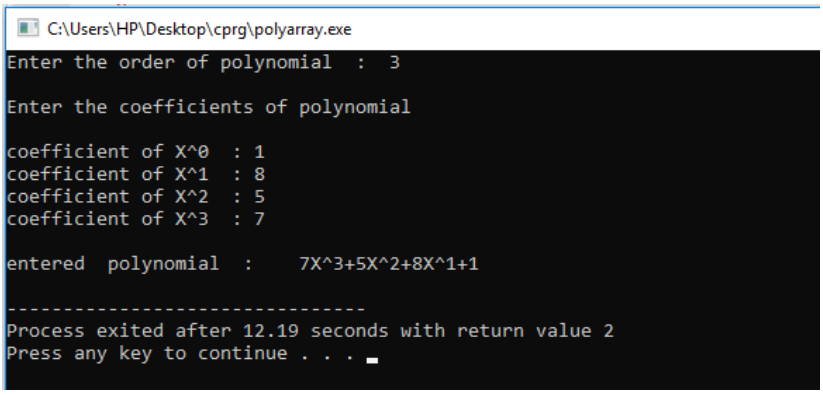


9. IMPLEMENT POLYNOMIAL USING ARRAYS

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

```
 int c[10];
int o,i;
printf("Enter the order of polynomial  :  ");
scanf("%d",&o);
printf("\nEnter the coefficients of polynomial\n\n");
for(i=0;i<=o;i++)
{
        printf("coefficient of X^%d  : ",i);
        scanf("%d",&c[i]);
}
        printf("\nentered  polynomial  :    ");
        for(i=o;i>0;i--)
        {
                if(c[i]!=0)

                        printf("%dX^%d+",c[i],i);

        }
        printf("%d\n",c[i]);
getch();
}
```

OUTPUT



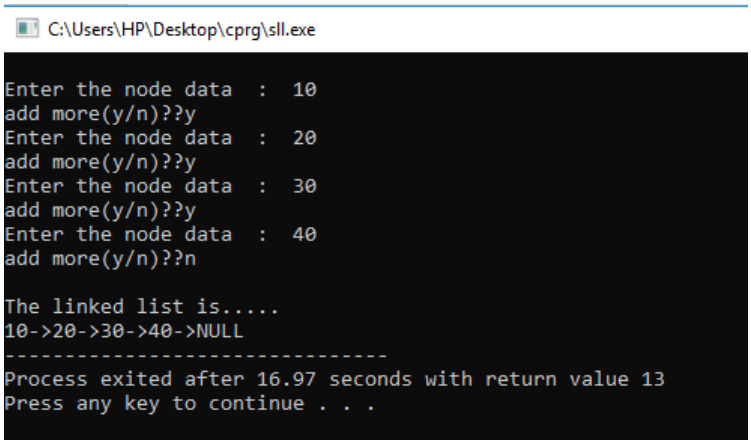
10. IMPLEMENT SINGLY LINKED LIST OF INTEGERS

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

```
#include<stdlib.h>
void main()
{
	struct node
	{
		int data;
		struct node *link;
	}*start=NULL,*new_node,*ptr;
char ch;
	do
	{
		new_node=(struct node*)malloc(sizeof(struct node));
		printf("\nEnter the node data  :  ");
		scanf("%d",&new_node->data);
		new_node->link=NULL;
		if(start==NULL)
		{
			start=new_node;
			ptr=new_node;
		}
		else
		{
			ptr->link=new_node;
			ptr=new_node;
		}
		printf("add more(y/n)??");
		ch=getche();
	}while(ch=='y'||ch=='Y');
	printf("\n\nThe linked list is.....\n");
	for(ptr=start;ptr!=NULL;ptr=ptr->link)
	printf("%d->",ptr->data);
	printf("NULL");
```

```
    getch();
}
```

OUTPUT



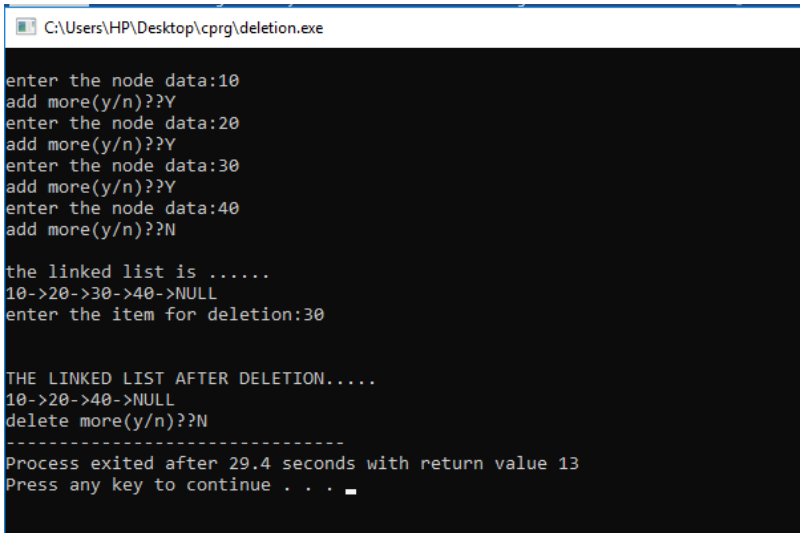
11. a) DELETION OF A ELEMENT FROM SINGLY LINKED LIST(Based on item)

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    struct node
    {
        int data;
        struct node*link;
    }*start=NULL,*new_node,*ptr;
    char ch;
    do
    {
    new_node=(struct node*)malloc(sizeof (struct node));
    printf("\nenter the node data:");
    scanf("%d",&new_node->data);
    new_node->link=NULL;
    if(start==NULL)
    {
        start=new_node;
        ptr=new_node;
```

```
        }
        else
        {
                ptr->link=new_node;
                ptr=new_node;
        }
        printf("add more(y/n)??");
        ch=getche();
}while(ch=='y'||ch=='Y');
printf("\n\nthe linked list is .....\n");
for(ptr=start;ptr!=NULL;ptr=ptr->link)
printf("%d->",ptr->data);
printf("NULL");
struct node *prev;
int item;
char ch1;
do
{
        if(start==NULL)
        {
                printf("\nmemory underflow-list is empty");
                exit(0);
        }
        else
        {
                printf("\nenter the item for deletion:");
                scanf("%d",&item);
                ptr=start;
                while(item!=ptr->data&&ptr!=NULL)
                {
                        prev=ptr;
                        ptr=ptr->link;
```

```
                    }
                    if(ptr==NULL)
                    printf("\nLOCATION ENTERED NOT EXIST");
                    else if(ptr==start)
                    {
                            start=ptr->link;
                            free(ptr);
                    }
                    else
                    {
                            prev->link=ptr->link;
                            free(ptr);
                    }
                    printf("\n\nTHE LINKED LIST AFTER DELETION.....\n");
                    for(ptr=start;ptr!=NULL;ptr=ptr->link)
                    printf("%d->",ptr->data);
                    printf("NULL");
            }
            printf("\ndelete more(y/n)??");
            ch1=getche();
    }while(ch1=='y'||ch1=='Y');
    getch();
}
```

OUTPUT



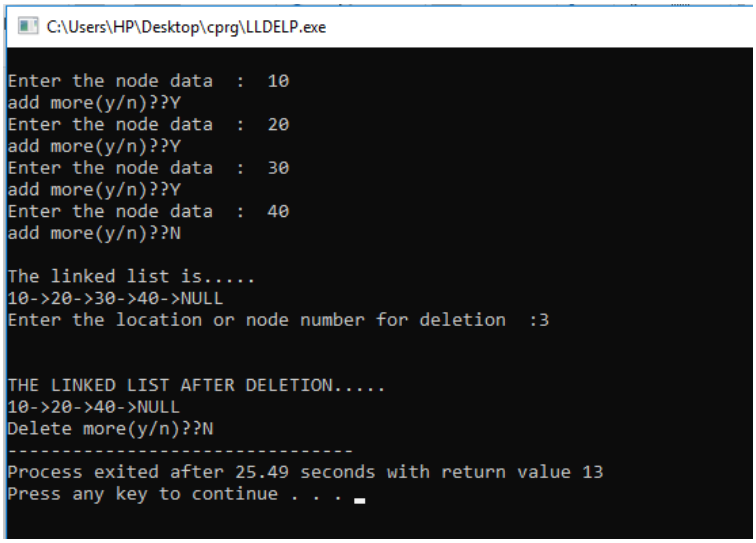
b) DELETION OF A ELEMENT FROM SINGLY LINKED LIST(Based on node position)

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
        struct node
        {
                int data;
                struct node *link;
        }*start=NULL,*new_node,*ptr;
char ch;
        do
        {
                new_node=(struct node*)malloc(sizeof(struct node));
                printf("\nEnter the node data  :  ");
                scanf("%d",&new_node->data);
                new_node->link=NULL;
                if(start==NULL)
                {
                        start=new_node;
                        ptr=new_node;
                }
                else
```

```
        {
                ptr->link=new_node;
                ptr=new_node;
        }
        printf("add more(y/n)??");
        ch=getche();
}while(ch=='y'||ch=='Y');
printf("\n\nThe linked list is.....\n");
for(ptr=start;ptr!=NULL;ptr=ptr->link)
printf("%d->",ptr->data);
printf("NULL");
struct node *prev;
int loc;
char ch1;
do
{
if(start==NULL)
{
        printf("\nMemory underflow-list is empty");
        exit(0);
}
else
{
        printf("\nEnter the location or node number for deletion  :");
        scanf("%d",&loc);
        ptr=start;
        int i=1;
        while(i<loc&&ptr!=NULL)
        {
                prev=ptr;
                ptr=ptr->link;
                i++;
```

```
                }
                if(ptr==NULL||loc<=0)
                printf("\nLOCATION ENTERED NOT EXIST");
                else if(ptr==start)
                {
                        start=ptr->link;
                free(ptr);
                }
                else
                {
                        prev->link=ptr->link;
                free(ptr);
                }
                printf("\n\nTHE LINKED LIST AFTER DELETION.....\n");
        for(ptr=start;ptr!=NULL;ptr=ptr->link)
        printf("%d->",ptr->data);
        printf("NULL");
}
        printf("\nDelete more(y/n)??");
                ch1=getche();
        }while(ch1=='y'||ch1=='Y');
        getch();
}
```

OUTPUT

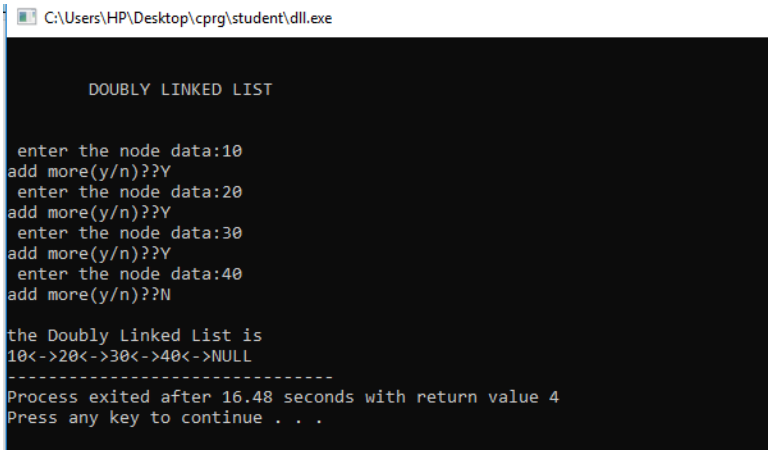


12. IMPLEMENT A DOUBLY LINKED LIST OF INTEGERS

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
        struct node
        {
                int data;
                struct node *link,*prev;
        }*start=NULL,*new_node,*ptr;
printf("\n\n\tDOUBLY LINKED LIST\n\n");
char ch;
        do
        {
                new_node=(struct node *)malloc(sizeof(struct node));
                printf("\n enter the node data:");
                scanf("%d",&new_node->data);
                new_node->link=NULL;
                if(start==NULL)
           {
                new_node->prev=NULL;
```

```
            start=new_node;
            ptr=new_node;
             }
             else
             {
                    ptr->link=new_node;
                    new_node->prev=new_node;
                    ptr=new_node;
             }
                    printf("add more(y/n)??");

                    ch=getche();
              }while(ch=='Y'||ch=='y');
             printf("\n\nthe Doubly Linked List is \n");
             for(ptr=start;ptr!=NULL;ptr=ptr->link)
             printf("%d<->",ptr->data);
             printf("NULL");
      }
```

OUTPUT



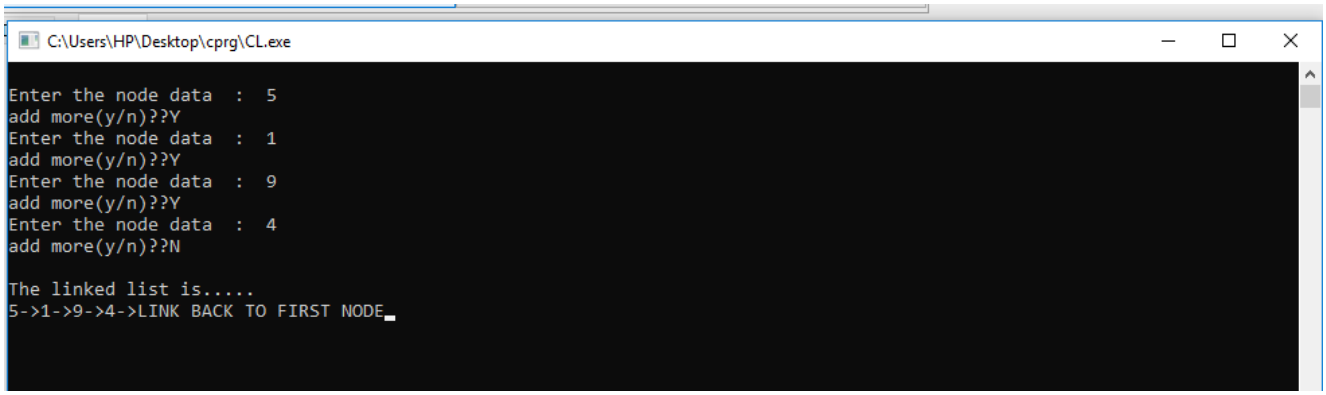
13. IMPLEMENTATION OF CIRCULAR LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>
void main()
```

```
{
        struct node
        {
                int data;
                struct node *link;
        }*start=NULL,*new_node,*ptr;
char ch;
        do
        {
                new_node=(struct node*)malloc(sizeof(struct node));
                printf("\nEnter the node data  :  ");
                scanf("%d",&new_node->data);
                new_node->link=start;
                if(start==NULL)
                {
                        start=new_node;
                        ptr=new_node;
                }
                else
                {
                ptr->link=new_node;
                ptr=new_node;
                }
                printf("add more(y/n)??");
                ch=getche();
        }while(ch=='y'||ch=='Y');
        printf("\n\nThe linked list is.....\n");
        ptr=start;
        do
        {
                printf("%d->",ptr->data);
                ptr=ptr->link;
```

```
        }while(ptr!=start);
        printf("LINK BACK TO FIRST NODE");
        getch();
}
```

OUTPUT

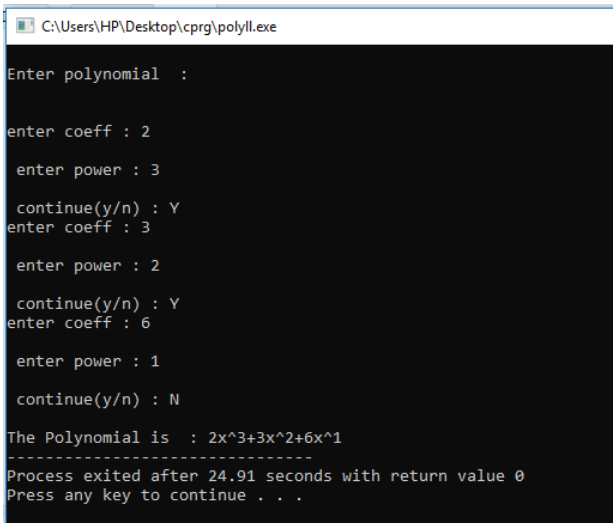


14. IMPLEMENT POLYNOMIAL USING LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>
struct link
{
      int coeff;
      int pow;
      struct link *next;
 }*poly=NULL;
void create(struct link *node)
{
 char ch;
 do
 {
  printf("\nenter coeff : ");
  scanf("%d",&node->coeff);
  printf("\n enter power : ");
  scanf("%d",&node->pow);
  node->next=(struct link*)malloc(sizeof(struct link));
```

```
  node=node->next;
  node->next=NULL;
  printf("\n continue(y/n) : ");
  ch=getche();
 }
 while(ch=='y' || ch=='Y');
}
void show(struct link *node)
{
 while(node->next!=NULL)
 {
  printf("%dx^%d",node->coeff,node->pow);
  node=node->next;
  if(node->next!=NULL)
   printf("+");
 }
}
void main()
{
      poly=(struct link *)malloc(sizeof(struct link));
      printf("\nEnter polynomial  : \n\n");
      create(poly);
      printf("\n\nThe Polynomial is  : ");
      show(poly);
      getch();
}
```

OUTPUT



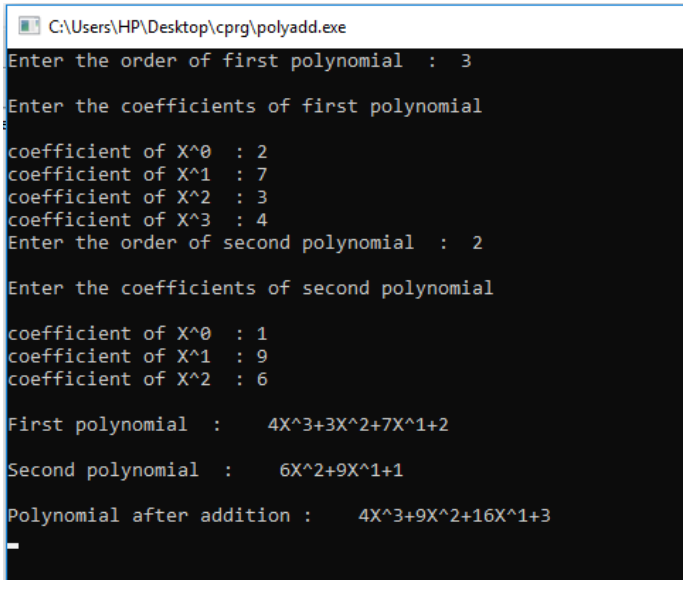
15. ADDITION OF 2 POLYNOMIALS

```
#include<stdio.h>
void main()
{
        int poly1[10],poly2[10],poly[10];
        int or1,or2,i;
        printf("Enter the order of first polynomial : ");
        scanf("%d",&or1);
        printf("\nEnter the coefficients of first polynomial\n\n");
        for(i=0;i<=or1;i++)
        {
                printf("coefficient of X^%d : ",i);
                scanf("%d",&poly1[i]);
        }
        printf("Enter the order of second polynomial : ");
        scanf("%d",&or2);
        printf("\nEnter the coefficients of second polynomial\n\n");
        for(i=0;i<=or2;i++)
        {
                printf("coefficient of X^%d : ",i);
                scanf("%d",&poly2[i]);
        }
```

```
                printf("\nFirst polynomial  :    ");
                for(i=or1;i>0;i--)
                {
                        if(poly1[i]!=0)

                                printf("%dX^%d+",poly1[i],i);

                }
                printf("%d\n",poly1[i]);
                        printf("\nSecond polynomial  :    ");
                for(i=or2;i>0;i--)
                {
                        if(poly2[i]!=0)
                        printf("%dX^%d+",poly2[i],i);

                }
                printf("%d\n",poly2[i]);
                //Adding the polynomials
if(or1>or2)
  {
                 for(i=0;i<=or2;i++)
                  {
                        poly[i]=poly1[i]+poly2[i];
                  }
                  for(i=or2+1;i<=or1;i++)
                  {
                        poly[i] = poly1[i];
                  }    }
else
 {
   for(i=0;i<=or1;i++)
```

```
    {
        poly[i]=poly1[i]+poly2[i];
    }
  for(i=or1+1;i<=or2;i++)
   {
    poly[i] = poly2[i];
   }
 }
                printf("\nPolynomial after addition :    ");
                for(i=or2+or1;i>0;i--)
                {
                        if(poly[i]!=0)
                        printf("%dX^%d+",poly[i],i);
                }
                printf("%d\n",poly[i]);
               getch();
}
```

OUTPUT



16. IMPLEMENT STACK USING ARRAY

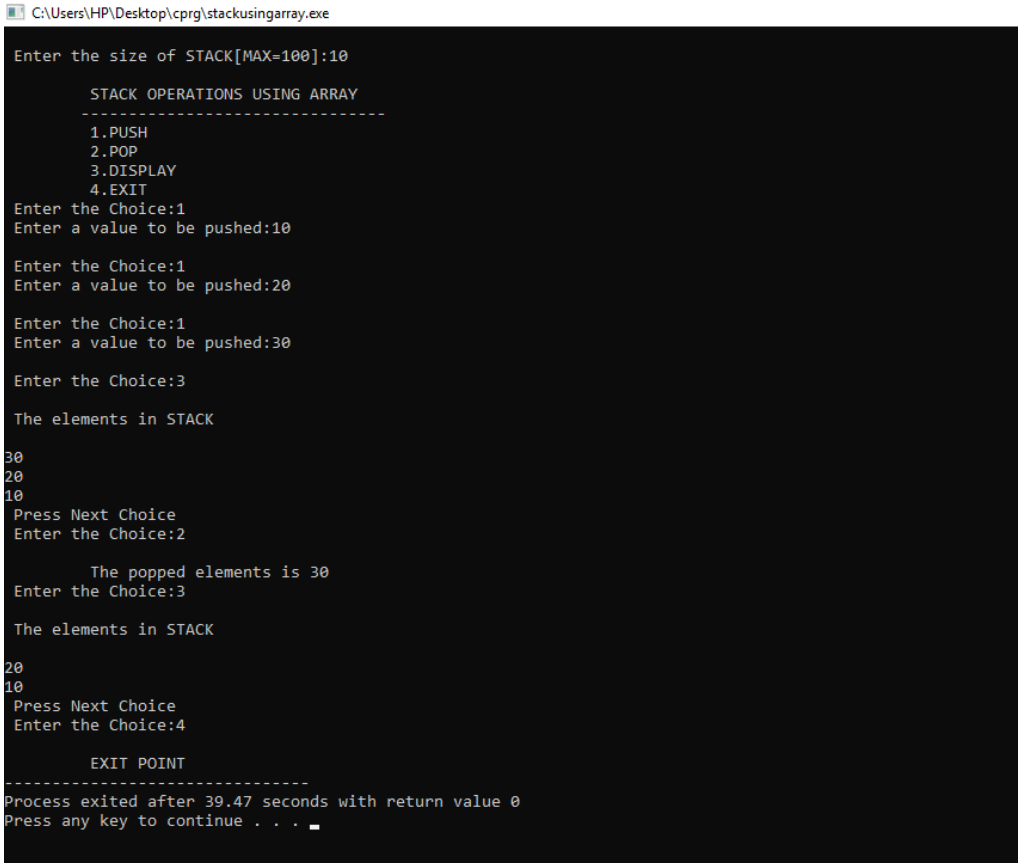
```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
```

```
void push();
void pop();
void display();
int main()
{
   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



17. IMPLEMENT STACK USING LINKED LIST

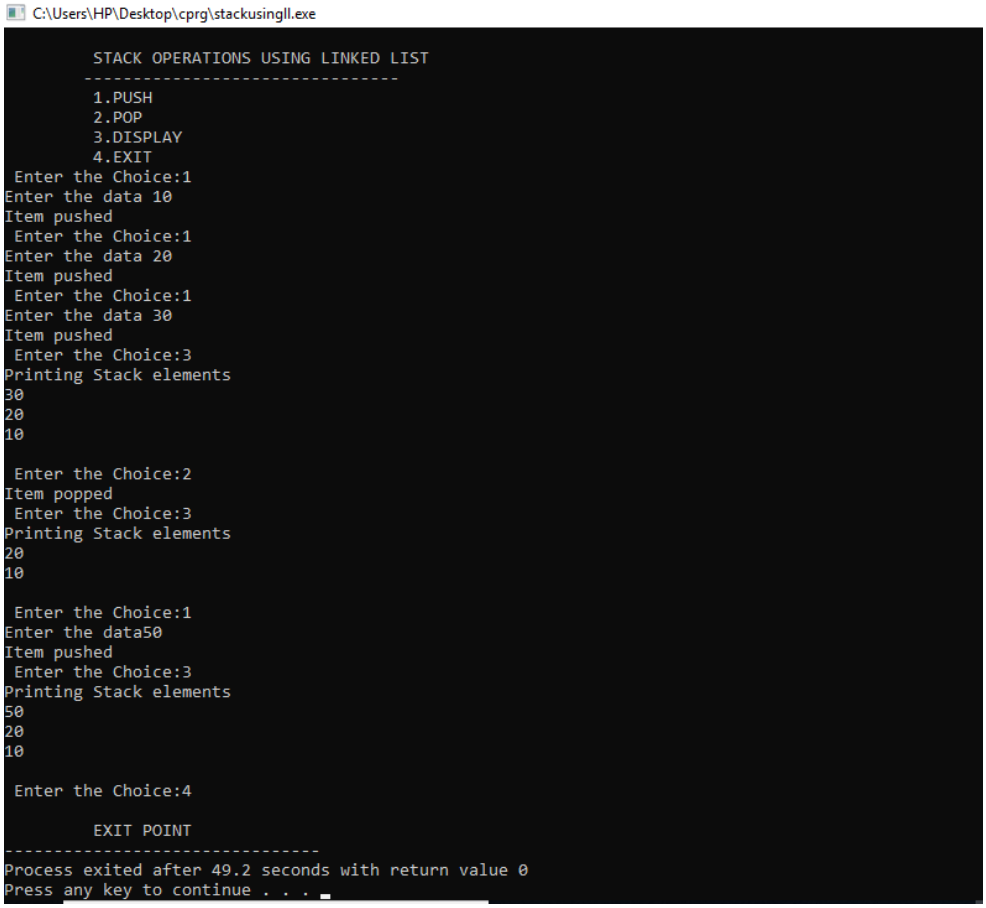
```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
int data;
struct node *next;
}*start;
  int main ()
{
    int choice=0;
    printf("\n\t STACK OPERATIONS USING LINKED LIST");
    printf("\n\t--------------------------------");
```

```
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 ()
{
    int data;
    struct node *newnode = (struct node*)malloc(sizeof(struct node));
          printf("Enter the data");
        scanf("%d",&data);
        newnode->data = data;
        if(start==NULL)
        {
            newnode -> next = NULL;
            start=newnode;
        }
        else
        {
           newnode ->next = start;
            start=newnode ;
        }
        printf("Item pushed");
}
void pop()
{
    int item;
    struct node *ptr;
    if (start == NULL)
    {
        printf("Underflow");
    }
```

```
    else
    {
        item = start->data;
        ptr = start;
        start = start->next;
        free(ptr);
        printf("Item popped");
    }
}
void display()
{
    int i;
    struct node *ptr;
    ptr=start;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->data);
            ptr = ptr->next;
        }
    }
}
```

OUTPUT



18. IMPLEMENT QUEUE USING ARRAY

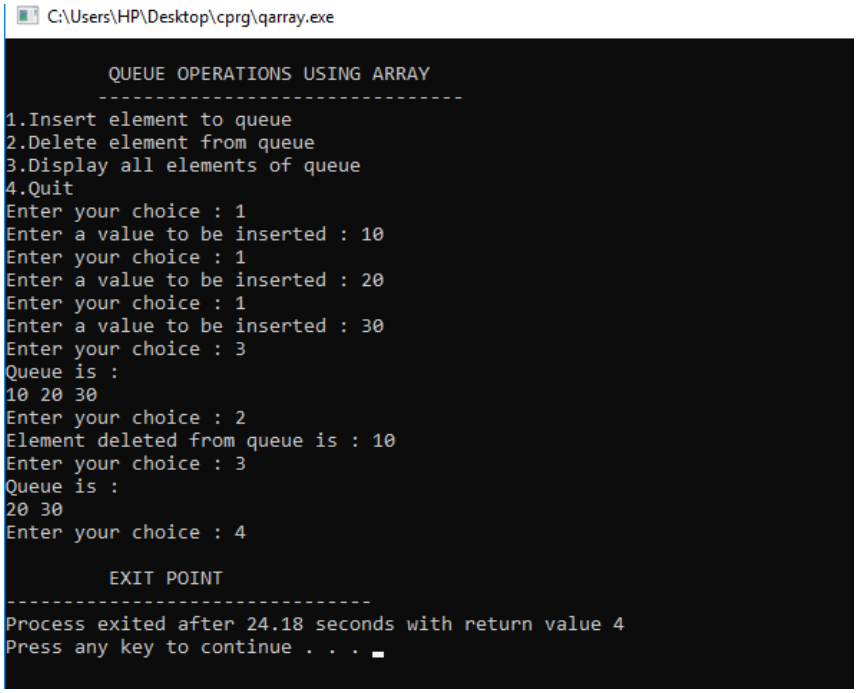
```
#include <stdio.h>
 void insert();
void delete();
void display();
int queue_array[100];
int rear = - 1;
int front = - 1;
main()
{
            int choice;
      printf("\n\t QUEUE OPERATIONS USING ARRAY");
    printf("\n\t--------------------------------\n");
       printf("1.Insert element to queue \n");
       printf("2.Delete element from queue \n");
```

```
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
         do
        {
        printf("Enter your choice : ");
        scanf("%d", &choice);
    switch (choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            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);
}

void insert()
{
    int add_item;
    if (rear == 99)
    printf("Queue Overflow \n");
    else
```

```
    {
        if (front == - 1)
        front = 0;
        printf("Enter a value to be inserted : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
}
 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;
    }
}
 void display()
{
    int i;
    if (front == - 1||front>rear)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
```

```
        printf("\n");
    }
}
```

OUTPUT



19. IMPLEMET QUEUE USING LINKED LIST

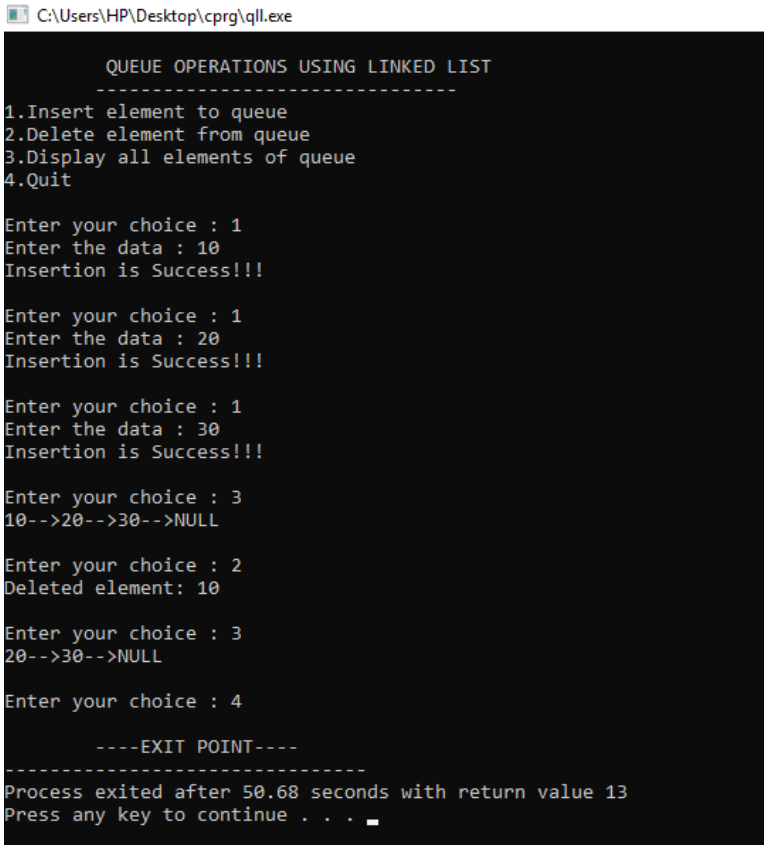
```
#include <stdio.h>
#include<stdlib.h>
 void insert();
void delete();
void display();
struct node
{
int data;
struct node *next;
}*front=NULL,*rear=NULL;
void main()
{
        int choice;
    printf("\n\t QUEUE OPERATIONS USING LINKED LIST");
  printf("\n\t--------------------------------\n");
```

```
        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");
         do
        {
        printf("\nEnter your choice : ");
        scanf("%d", &choice);
       switch (choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
             printf("\n\t----EXIT POINT---- ");
                break;
            default:
             printf ("\nPlease Enter a Valid Choice(1/2/3/4)");
        }
    }while(choice!=4);
    getch();
}
 void insert()
{
             int data;
  struct node *newnode = (struct node*)malloc(sizeof(struct node));
```

```
    printf("Enter the data : ");
    scanf("%d",&data);
   newnode->data = data;
   newnode -> next = NULL;
   if(front == NULL)
      front = rear = newnode;
   else
   {
      rear -> next = newnode;
      rear = newnode;
   }
   printf("Insertion is Success!!!\n");
}
void delete()
{
   if(front == NULL)
      printf("\nQueue is Empty!!!");
   else
   {
      struct node *temp = front;
      front = front -> next;
      printf("Deleted element: %d\n", temp->data);
      free(temp);
   }
}
void display()
{
   if(front == NULL)
      printf("\nQueue is Empty!!!\n");
   else
   {
      struct node *temp = front;
```

```
    while(temp->next != NULL)
            {
            printf("%d-->",temp->data);
            temp = temp -> next;
    }
    printf("%d-->NULL\n",temp->data);
  }
}
```

OUTPUT



20. BUBBLE SORT

```
#include <stdio.h>
void main()
{
   int  n, i, j,temp;
   int array[100];
```

```
    printf("\tBUBBLE SORT\n.....................\n");
    printf("Enter number of elements(between 1 & 100) ");
    scanf("%d", &n);
    printf("\nEnter %d integers \n", n);
    for(i = 0; i < n; i++)
    {
       scanf("%d",&array[i]);
    }
          for(i=0;i<n-1;i++)
                  for(j=0;j<(n-i-1);j++)
                          if(array[j]>array[j+1])
                          {
                                  temp=array[j];
                                  array[j]=array[j+1];
                                  array[j+1]=temp;
                          }
          printf("\nsorted array\n");
          for(i = 0; i < n; i++)
                  printf("%d\t",array[i]);
   getch();
}
```

OUTPUT

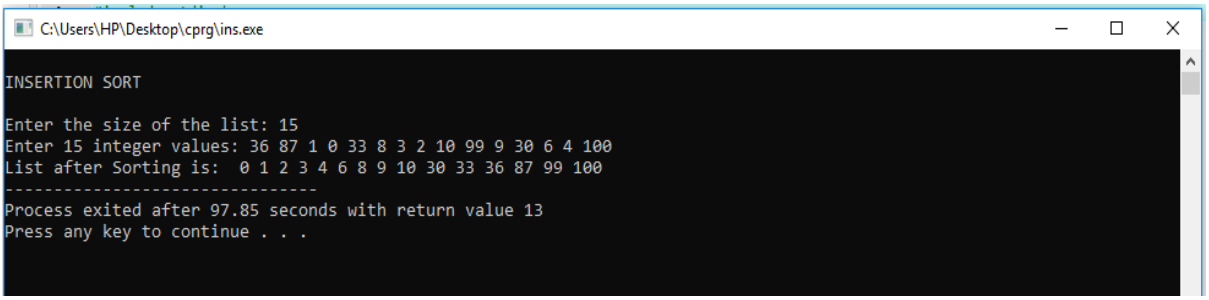
```
        BUBBLE SORT
.....................
Enter number of elements(between 1 & 100) 7

Enter 7 integers
10 49 23 67 11 7 1

sorted array
1       7       10      11      23      49      67
```

21. INSERTION SORT

```
#include<stdio.h>

void main()

{
```

```
  int n, i, j, temp, array[100];
  printf("\nINSERTION SORT\n\n");
  printf("Enter the size of the list: ");
  scanf("%d", &n);
   printf("Enter %d integer values: ", n);
  for (i = 0; i < n; i++)
    scanf("%d", &array[i]);
  //Insertion sort logic
  for (i = 1; i < n; i++)
   {
    temp = array[i];
    j = i - 1;
    while ((temp < array[j]) && (j >= 0))
    {
      array[j + 1] = array[j];
      j = j - 1;
    }
    array[j + 1] = temp;
  }
  printf("List after Sorting is: ");
  for (i = 0; i < n; i++)
    printf(" %d", array[i]);
 getch();
}
```

OUTPUT

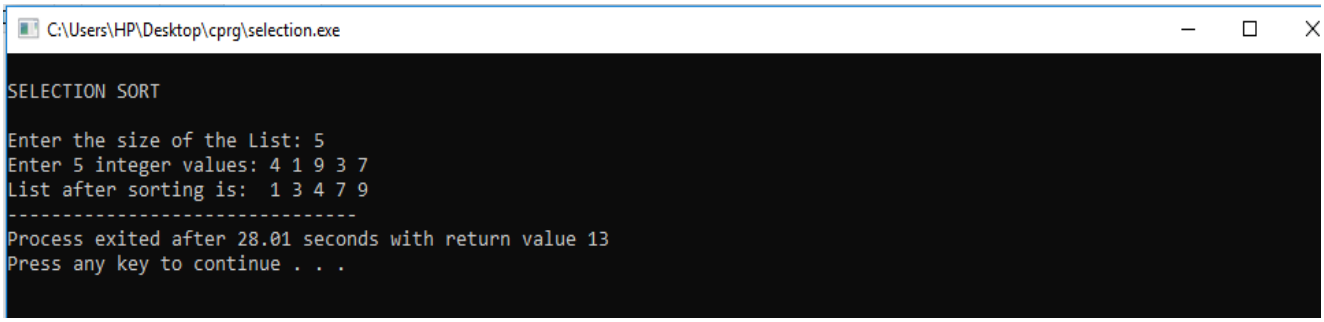


22. SELECTION SORT

```
#include<stdio.h>
void main()
{
  int n,i,j,temp,array[100];
  printf("\nSELECTION SORT\n\n");
  printf("Enter the size of the List: ");
  scanf("%d",&n);
  printf("Enter %d integer values: ",n);
  for(i=0; i<n; i++)
     scanf("%d",&array[i]);
   //Selection sort logic
  for(i=0; i<n; i++)
  {
     for(j=i+1; j<n; j++)
            {
         if(array[i] > array[j])
             {
             temp=array[i];
             array[i]=array[j];
             array[j]=temp;
```

```
        }
    }
  }
  printf("List after sorting is: ");
  for(i=0; i<n; i++)
     printf(" %d",array[i]);
  getch();
}
```

OUTPUT



23. MERGE SORT

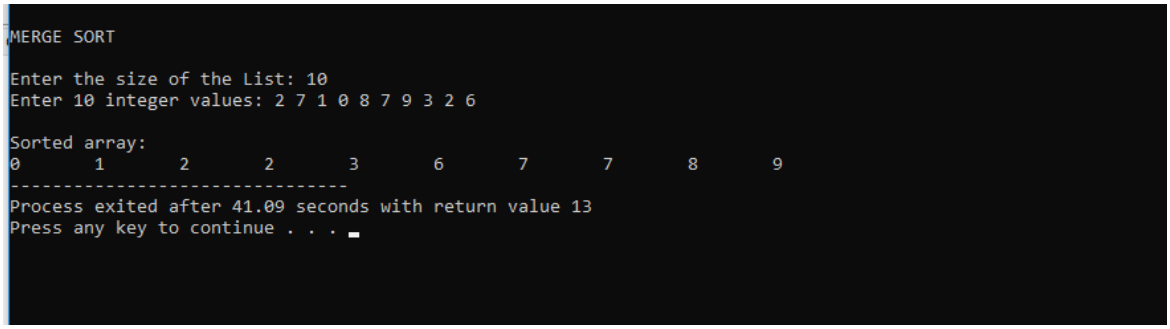
```
#include <stdio.h>
void merge(int a[], int p, int q, int r)
{
   int b[100];
   int i, j, k;
   k = 0;
   i = p;
   j = q + 1;
   while(i <= q && j <= r)
   {
      if(a[i] < a[j])
```

```
    {
     b[k]=a[i];
              k++;
              i++;
    }
    else
    {
       b[k]=a[j];
                k++;
                      j++;
    }
}

while(i <= q)
{
    b[k]=a[i];
              k++;
               i++;
}
  while(j <= r)
{
   b[k]=a[j];
      k++;
  j++;
}
```

```
for(i=r; i >= p; i--)
    {
     --k;
       a[i] = b[k];
    }
 }
 void mergeSort(int a[], int p, int r)
 {
    int q;
    if(p < r)
    {
       q = (p + r) / 2;
       mergeSort(a, p, q);
       mergeSort(a, q+1, r);
       merge(a, p, q, r);
    }
 }
 void main()
 {
   int n,arr[100],i;
  printf("\nMERGE SORT\n\n");
  printf("Enter the size of the List: ");
  scanf("%d",&n);
  printf("Enter %d integer values: ",n);
  for(i=0; i<n; i++)
```

```
        scanf("%d",&arr[i]);
    mergeSort(arr, 0, n-1);
    printf("\nSorted array: \n");
    for(i=0; i<n; i++)
        printf("%d\t",arr[i]);
    getch();
  }
```

OUTPUT



24. QUICK SORT

```
#include<stdio.h>
void quickSort(int [],int,int);
int main()
{
  int list[100],size,i;
printf("\nQUICK SORT\n\n");
  printf("Enter size of the list: ");
  scanf("%d",&size);
  printf("Enter %d integer values: ",size);
  for(i = 0; i < size; i++)
    scanf("%d",&list[i]);
```

```
  quickSort(list,0,size-1);
  printf("List after sorting is: ");
  for(i = 0; i < size; i++)
    printf(" %d",list[i]);
  return 0;
}
void quickSort(int list[],int first,int last)
{
    int pivot,i,j,temp;
     if(first < last)
{
         pivot = first;
         i = first;
         j = last;
  while(i < j)
 {
     while(list[i] <= list[pivot] && i < last)
                i++;
      while(list[j] > list[pivot])
                j--;
        if(i <j)
        {
                 temp = list[i];
                 list[i] = list[j];
                 list[j] = temp;
```

```
            }
        }
        temp = list[pivot];
        list[pivot] = list[j];
        list[j] = temp;
        quickSort(list,first,j-1);
        quickSort(list,j+1,last);
    }
}
```

OUTPUT: