```
/*
 * Q-1.c
 *
 *  Created on: 26-Jul-2024
 *      Author: root
 */

//Write a program to match two different
pointer in 1 location in array

#include<stdio.h>

int main()
{
   int n;

   printf("Enter The Array Size : ");
      scanf("%d",&n);

   int a[n];

      int *p1;
      int *p2;

      p1=&a[0];
      p2=&a[n-1];

      for(int i=0 ;i<n ;i++)
      {
         printf("Enter The Number [%d] :
",i+1);
         scanf("%d",p1+i);

      }
```

```
        for(int i=0 ;i<n ;i++)
        {
           printf("%d ",*(p1+i));

        }
        while(p1!=p2)
        {
           p1++;
           p2--;
        }
        printf("\n This is Pointer Value
%d  \n",*p1);
        printf(" This is the same Location Of
The Pointer %u %u",p2,p1);

    return 0;
}

/*
 * Q-2.c
 *
 *  Created on: 26-Jul-2024
 *      Author: root
 */

//Implement a program to find length of a
string with pointer

#include<stdio.h>
#include<string.h>
int main()
{
    int n,lenth=0;

        printf("Enter THe string Size : ");
```

```
        scanf("%d",&n);

    char str[n];

    char *p;

    p=str;

        printf("Enter The string : ");
        scanf("%s",p);

        int i=0;

        while(p[i]!='\0')
        {
            lenth++;
            i++;
        }

        printf("\nYour String : %s\n",p);
        printf("Your String Size :
%d",lenth);




        return 0;
}


/*
 * Q-3.c
 *
 *  Created on: 27-Jul-2024
 *      Author: root
```

```
 */


//Write a program to find min and max
value from array with pointers

#include<stdio.h>

int main()
{
   int n;

       printf("Enter The array Size : ");
          scanf("%d",&n);

       int a[n];
       int *p;

          for(int i =0 ; i<n ; i++)
          {
             printf("\nEnter THe value :
[%d]",i+1);

                scanf("%d",&a[i]);
          }

       p=a;

          int max=*p;

             for(int i=0 ; i<n ;i++)
             {
                if(*(p+i)>max)
                {
                   max=*(p+i);
```

```
            }
        }

        printf("\nMax value = %d ",max);
}



/*
 * Q-4.c
 *
 *  Created on: 27-Jul-2024
 *      Author: root
 */

//Write a program to perform various
testing on pointerseg. *p++, p--, *++p



#include<stdio.h>

int main()
{
   int n;

      printf("Enter THe array Size : ");
         scanf("%d",&n);

      int a[n];
      int *p;

         for(int i=0 ;i<n ;i++)
         {
            scanf("%d",&a[i]);
```

```
        }

        p=&a[0];

           for(int i=0 ; i<n ; i++)
           {
              printf("\n *p++ %d\n",*p++);

           }

           for(int i=0 ; i<n ; i++)
           {
              printf("\np-- %u\n",p--);
           }

           for(int i=0 ; i<n ; i++)
           {
              printf("\n*++p %d\n",*++p);
           }

           return 0;

}



/*
 * Q-5.c
 *
 *  Created on: 27-Jul-2024
 *      Author: root
 */
```

```
//Write a program to implement student
structure and display student detail in
descending
//order of their SGPA

#include<stdio.h>

struct student{

   char name[10];
   int Roll_no;
   float SGPA;

};

int main()
{
   int n;
      printf("Enter The Structure Size : ");
         scanf("%d",&n);

   struct student s[n];

   for(int i=0 ; i<n ; i++)
   {

      printf("\nEnter The Roll No :");
         scanf("%d",&s[i].Roll_no);

      printf("Enter The Name : ");
         scanf("%s",s[i].name);

      printf("Enter The SGPA : \n");
         scanf("%f",&s[i].SGPA);
```

```
    }

    printf("Roll No\t Name\t SGPA\n");
          for(int i=0 ; i<n ; i++)
          {
             printf("\n  %d\t %s\t
%f\t",s[i].Roll_no,s[i].name,s[i].SGPA);

          }


          struct student temp;

             for(int i=0 ; i<n-1 ;i++)
             {
                for(int j=i+1 ; j<n ;j++)
                {
                   if(s[i].SGPA<s[j].SGPA)
                   {
                      temp=s[j];
                      s[j]=s[i];
                      s[i]=temp;
                   }
                }

             }

             printf("\nRoll No\t Name\t
SGPA\n");
                      for(int i=0 ; i<n ; i++)
                      {
                         printf("\n  %d\t %s\t
%f\t",s[i].Roll_no,s[i].name,s[i].SGPA);

                      }
```

```
    return 0;

}




/*
 * Q-6.c
 *
 *  Created on: 27-Jul-2024
 *      Author: root
 */

/*
 * Q-5.c
 *
 *  Created on: 27-Jul-2024
 *      Author: root
 */

//Write a program to implement student
structure and display student detail in
descending
//order of their SGPA

#include<stdio.h>

struct student{

    char name[10];
    int Roll_no;
    float SGPA;

};
```

```
int main()
{
   int n;
      printf("Enter The Structure Size : ");
         scanf("%d",&n);

   struct student s[n];

   struct student *str=&s[0];

   for(int i=0 ; i<n ; i++)
   {

      printf("\nEnter The Roll No :");
         scanf("%d",&(str+i)->Roll_no);

      printf("Enter The Name : ");
         scanf("%s",(str+i)->name);

      printf("Enter The SGPA : \n");
         scanf("%f",&(str+i)->SGPA);

   }

   printf("Roll No\t Name\t SGPA\n");
         for(int i=0 ; i<n ; i++)
         {
            printf("\n  %d\t %s\t %f\t",(str+i)-
>Roll_no,(str+i)->name,(str+i)->SGPA);

         }

         struct student temp;
```

```
            for(int i=0 ; i<n-1 ;i++)
            {
               for(int j=i+1 ; j<n ;j++)
               {
                  if((str+i)->SGPA<(str+j)-
>SGPA)
                  {
                     temp=str[j];
                     str[j]=str[i];
                     str[i]=temp;
                  }
               }

            }

            printf("\nRoll No\t Name\t
SGPA\n");
                     for(int i=0 ; i<n ; i++)
                     {
                        printf("\n  %d\t %s\t
%f\t",(str+i)->Roll_no,(str+i)->name,(str+i)-
>SGPA);

                     }
   return 0;

}



/*
 * Q-7.c
 *
 *  Created on: 27-Jul-2024
 *      Author: root
```

```
 */

//Write a program to perform all
operations on 1D and 2D array

#include<stdio.h>

int size = 5;
#define ROWS 3
#define COLS 3

// 1D Array
void input_1Darry(int a[])
{
      for(int i=0 ; i<size ; i++)
      {
         printf("\nEnter The Value [%d] : ",i);
         scanf("%d",&a[i]);
      }
}

void Display_1Darry(int a[])
{
         for(int i=0 ; i<size ; i++)
         {
            printf(" %d ",a[i]);
         }
}
void sort_1Darry(int a[])
{
   int i,j,temp=0;
   printf("\n1d Array Assending Sort ...\n");

      for( i=0 ,j=size-1 ;i<j;i++,j--)
      {
```

```
            temp=a[j];
            a[j]=a[i];
            a[i]=temp;
        }

            for(int i=0 ; i<size ; i++)
               {
                  printf("%d ",a[i]);
               }

}

void Search_1Darry(int a[])
{

   int se,flag=0;

   printf("\nEnter The value you Want to
Search  : ");
        scanf("%d",&se);

       for(int i=0 ; i<size ; i++)
       {
          if(a[i]==se)
          {
             flag=1;
             printf("\nNumber Is Available ");
             printf("\nNumber index %d ",i);

          }
       }
       if(flag==0)
       {
          printf("\nNumber Is Not Available
");
```

```
        }

}

void Delete_1Darry(int a[])
{
   int flag=0;

   int delete;

       printf("\n Enter Your Number You
Want To delete : ");
          scanf("%d",&delete);

          for(int i=0 ; i<size;)
          {
             if(a[i] == delete)
             {
                for(int j=i ; j<size ; j++)
                {
                   a[j]=a[j+1];
                }
                size=size-1;
                flag=1;

                printf("\n Number is Deleted ");
             }
             else
             {
                i++;
             }
          }
          if(flag==0)
          {
```

```
            printf("\n Number Is Not
Available : ");
        }
        printf("\n After Deletion Array
Elements :\n");
}




//2D Array

void input_2Darry(int arr[ROWS][COLS])
{
   for(int i=0 ; i<ROWS ; i++)
   {
      for(int j=0 ; j<COLS ; j++)
      {
         printf("\nEnter 2D Array
Elements  [%d][%d] : ",i,j);

            scanf("%d",&arr[i][j]);
      }
   }
}

void Display_2Darry(int arr[ROWS][COLS])
{
   for(int i=0 ; i<ROWS ; i++)
   {
      for(int j=0 ; j<COLS ; j++)
      {
            printf("%d " ,arr[i][j]);
      }
```

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

void short_2Darry(int arr[ROWS][COLS])
{
    for(int i=0 ; i<ROWS ; i++)
        {
            for(int j=0 ; j<COLS ; j++)
            {
                for(int k=0 ; k<ROWS ; k++)
                {
                    for(int l=0 ; l<COLS ; l++)

                    {
                        if(arr[i][j]>arr[k][l])
                        {
                        int temp=arr[i][j];
                        arr[i][j]=arr[k][l];
                        arr[k][l]=temp;
                        }
                    }
                }
            }
        }

}


void Search_2Darry(int arr[ROWS][COLS])
{

    int se,flag=0;
```

```
    printf("\nEnter The value you Want to
Search  : ");
        scanf("%d",&se);

       for(int i=0 ; i<ROWS ; i++)
       {
          for(int j=0 ;j<ROWS ; j++)
          {
             if(arr[i][j]==se)
             {
                flag=1;
                printf("\nNumber Is Available
");
                printf("\nNumber index %d
%d",i,j);

             }
          }

       }
       if(flag==0)
       {
          printf("\nNumber Is Not Available
");

       }

}


int main()
{
   int choice;

   int a[size];
```

```
menu:

        printf("\n 1. Performance 1D
Array\n");
        printf(" 2. Performance 2D
Array\n");
        printf(" 3. Exit..\n");

        printf("\n--Enter Your Choice : ");
          scanf("%d",&choice);

          switch(choice)
          {
            case 1:

              menu1:

                  printf("\n 1. Input : \n");
                  printf(" 2. Display\n");
                  printf(" 3. Short\n");
                  printf(" 4. Search\n");
                  printf(" 5. Delete\n");
                  printf(" 6. Goto Main
Menu\n");

                int sub1choice;

                  printf("\n--Enter The
Choice : ");
                    scanf("%d",&sub1choi
ce);

                switch(sub1choice)
```

```
{
   case 1:
         input_1Darry(a);

         goto menu1;

   case 2:
         Display_1Darry(a);

         goto menu1;

   case 3:
         sort_1Darry(a);

         goto menu1;

   case 4:

         Search_1Darry(a);

         goto menu1;

   case 5:
      Delete_1Darry(a);
      Display_1Darry(a);

         goto menu1;

   case 6:

         goto menu;
         break;

   default :
```

```
                printf("\nPlease !
Enter The Valid Choice ...");

                }

        case 2:

            menu2:

                printf("\n 1. Input :
\n");
                printf(" 2. Display\n");
                printf(" 3. Short\n");
                printf(" 4. Search\n");
                printf(" 5. Delete\n");
                printf(" 6. Goto Main
Menu\n");

                   int sub2choice;

                   int
arr2D[ROWS][COLS];


                printf("\n--Enter The
Choice : ");
                     scanf("%d",&sub2
choice);

                   switch(sub2choice)
                   {
                   case 1:

                        input_2Darry(ar
r2D);
```

```
                goto menu2;

            case 2:
                Display_2Darry(
arr2D);

                goto menu2;

            case 3:
                short_2Darry(arr2
D);

                Display_2Darry(ar
r2D);

                goto menu2;

            case 4:
                Search_2Darry(ar
r2D);
                goto menu2;

//              case 5:
//                  Delete_2Darry(a
rr2D);
//
//                  goto menu2;



            default :

                printf("\nPlease !
Enter The Valid Choice ...");
```

```
                        }

            case 3:

                printf("Exit This Program
...\n");
                break;

            default :
                printf("\n Please ! Enter The
Valid Choice...");

        }









   return 0;
}


/*
 * Q-8.c
 *
 *  Created on: 29-Jul-2024
 *      Author: root
 */

#include<stdio.h>

int main()
```

```
{
   int arr[2][2][2];


   for(int i=0 ; i<2 ; i++)
   {
      for(int j=0 ; j<2;j++)
      {
         for(int k=0 ; k<2 ; k++)
         {
            printf("\nEnter The Student
Marks  : [%d][%d][%d] ",i,j,k);
            scanf("%d",&arr[i][j][k]);
         }
      }
   }

   for(int i=0 ; i<2 ; i++)
      {
         for(int j=0 ; j<2 ;j++)
         {
            for(int k=0 ; k<2 ; k++)
            {
               printf("%d ",arr[i][j][k]);
            }
//             printf("\n");
         }

         printf("\n");

      }

   return 0;

}
```

```
/*
 * Q-9.c
 *
 *  Created on: 29-Jul-2024
 *      Author: root
 */

#include<stdio.h>

void array1()
{
   int n;

       printf("Enter The array size : ");
          scanf("%d",&n);

   int a1[n];

       for(int i=0 ; i<n ;i++)
       {
          printf("Enter The Elements [%d] :
",i);

              scanf("%d",&a1[i]);
       }

       printf("\n       Elements    Address\n\n
");

       for(int i=0 ;i<n; i++)
       {
          printf("Arry[%d]\t   %d\t   %u\n",i
,a1[i] , &a1[i]);
```

```
        }


}

void array2()
{
   int x,y;

          printf("\nEnter The 2D Array Size :
");
             scanf("%d %d",&x,&y);

             int a2[x][y];

             for(int i=0 ; i<x ; i++)
             {
                for(int j=0 ; j<y ; j++)
                {
                   printf("Enter The Elemets
[%d][%d] : ",i,j);

                      scanf("%d",&a2[i][j]);
                }
             }

             printf("\n\t    Elements\t
Address\n\n");

                for(int i=0 ; i<x ; i++)
                {
                   for(int j=0 ; j<y ; j++)
                   {
                      printf("Arry[%d][%d]\t
%d\t %u\n",i,j,a2[i][j] , &a2[i][j]);
```

```
                }
            }

}

int main()
{

   int choice;

   menu:

        printf("\n 1. 1D Array Performance
\n");
        printf(" 2. 2D Array Performance
\n");
        printf(" 3. Exit....");

      printf("\n---Enter Your Choice : ");
        scanf("%d",&choice);

        switch(choice)
        {
        case 1:
          array1();
          goto menu;

        case 2:
                array2();
                goto menu;
        case 4:

          printf("Exit Program : ");

          break;
```

```
        default :
                printf("Enter The Valid Choice :
");

        }

        return 0;
}


#include <stdio.h>

// Function Declaration

// glbal variable Declaration

int top = -1;
int stack[20];

// main function

// Function Definition For Push Operation

void Push(int value, int size)
{
   if (top == (size - 1))
   {
      printf("\n----Stack Overflow----\n");
   }

   else
   {
      top = top + 1;

      stack[top] = value;
```

```
    }
}

// Function Definition For Pop Operation

void Pop()
{
    int value;

    if (top == -1)
    {
        printf("\n----Stack Underflow----\n");
    }

    else
    {
        value = stack[top];
        top = top - 1;

        printf("%d", value);
    }
}

// Function Definition For Pop Operation

void Display()
{
    int i;

    if (top == -1)
    {
        printf("\n----Stack Is Underflow \n");
    }
```

```
    printf("\n----Stack Elements are Below :
\n");

    for (i = top; i >=0; i--)
    {
        printf("\n Elemets = %d \n ", stack[i]);
    }
}

// Function Definition For Pop Operation

void Peek()
{
    if (top == -1)
    {
        printf("\n----Stack Is Empty----\n");
    }

    else
    {
        printf("\n Peeked Element = %d",
stack[top]);
    }
}

void Peep()
{
    int k,i;

    if(top == -1)
    {
        printf("\n----Stack Is Underflow----
\n");
    }
```

```
    else
    {
        printf("\n Enter Your Peep Elemets :
");
            scanf("%d",&k);
    }

    if(k>top+1)
    {
        printf("\n--Peep Is Not Found !--");
    }
    else
    {
        printf(" Peeped Elements %d :
",stack[top-k+1]);
    }

}

void Top_to_Bottom()
{
    int i;

    for (i = top; i >=0; i--)
    {
        printf("\n Elemets = %d \n ", stack[i]);
    }
}

void Bottom_to_top()
{
    int i;

    for (i = 0; i<top; i++)
    {
```

```
        printf("\n Elemets = %d \n ", stack[i]);
    }
}


int main()
{
    int ch, size, value;

    printf("Enter Size Of Stack : ");
    scanf("%d", &size);

    // menu:

menu:

    printf("\n 1. Push\n");
    printf(" 2. Pop\n");
    printf(" 3. Display\n");
    printf(" 4. Peek\n");
    printf(" 5. Peep\n");
    printf(" 6. Top_to_Bottom\n");
    printf(" 7. Bottom_to_top\n");
    printf(" 8. Exit !...\n");

    // input choice from user

    printf("\n Enter Your Choice : ");
    scanf("%d", &ch);

    // switch case

    switch (ch)
    {
    case 1:
```

```
        printf("\nEnter Element To be Pushed:
");
        scanf("%d", &value);
        Push(value, size);
        Display();

       goto menu;

    case 2:

        Pop();
        goto menu;

    case 3:

        Display();
        goto menu;

    case 4:

        Peek();
        goto menu;

    case 5:

        Peep();

        goto menu;

    case 6:
        Top_to_Bottom();

        goto menu;
```

```
    case 7:

        Bottom_to_top();

        goto menu;
    case 8:

        printf("Exit The Program Thank you
!...");

    default:

        printf(" Please Enter The Valid Choice :
");
    }

    return 0;
}
#include <stdio.h>

// Function Declaration

// glbal variable Declaration

int top = -1;
int stack[20];

// main function

// Function Definition For Push Operation

void Push(int value, int size)
{
    if (top == (size - 1))
    {
```

```
        printf("\n----Stack Overflow----\n");
    }

    else
    {
        top = top + 1;

        stack[top] = value;
    }
}

// Function Definition For Pop Operation

void Pop()
{
    int value;

    if (top == -1)
    {
        printf("\n----Stack Underflow----\n");
    }

    else
    {
        value = stack[top];
        top = top - 1;

        printf("%d", value);
    }
}

// Function Definition For Pop Operation

void Display()
{
```

```
    int i;

    if (top == -1)
    {
        printf("\n----Stack Is Underflow \n");
    }

    printf("\n----Stack Elements are Below :
\n");

    for (i = top; i >= 0; i--)
    {
        printf("\n Elemets = %d \n ", stack[i]);
    }
}

int main()
{
    int ch, size, value;

    printf("Enter Size Of Stack : ");
    scanf("%d", &size);

    // menu:

menu:

    printf("\n 1. Push\n");
    printf(" 2. Pop\n");
    printf(" 3. Display\n");

    printf(" 4. Exit !...\n");

    // input choice from user
```

```
    printf("\n Enter Your Choice : ");
    scanf("%d", &ch);

    // switch case

    switch (ch)
    {
    case 1:

        printf("\nEnter Element To be Pushed:
");
        scanf("%d", &value);
        Push(value, size);
        Display();

        goto menu;

    case 2:

        Pop();
        goto menu;

    case 3:

        Display();
        goto menu;

    case 4:

        printf("Exit The Program Thank you
!...");

    default:
```

```
        printf(" Please Enter The Valid Choice :
");
    }

    return 0;
}

/*
 * Q-12.c
 *
 *  Created on: 03-Aug-2024
 *      Author: root
 */


#include<stdio.h>

//global Variable

int top=-1;
int stack[20];
int *ptr=&stack[0];

//function Declaration;

void Push(int value,int size);
void Pop();
void Display();


//Function Push

void Push(int value,int size)
{
    int *vl;
```

```
    vl=&value;

    if(top==(size-1))
    {
        printf("\n----Stack Is Overflow----\n");
    }

    else
    {
        top=top+1;
        *ptr=*vl;
        ptr++;
        vl++;
    }
}

//Function Pop

void Pop()
{
    int value;
    int *vl;
    vl=&value;

        if(top==-1)
        {
            printf("\n----Stack Underflow----
\n");
        }
        else
        {
            *vl=*ptr;
            top=top-1;
                printf("%d",*vl);
        }
```

```
}

//Function Display

void Display()
{
   int i;

       if(top==-1)
       {
          printf("\n----Stack Is Underflow----
\n");
       }

          printf("\n--STack Elements are
Below--\n");

          for(i=top ; i>=0 ; i--)
          {
             printf("%d\n",*ptr);
             ptr++;
          }
}



int main()
{
   int ch,size,value;
   int *vl;
   vl=&value;

       printf("Enter Size Of Stack : ");
          scanf("%d",&size);
```

```
menu:

    printf("\n 1. Push\n");
    printf(" 2. Pop\n");
    printf(" 3. Display\n");
    printf(" 4. Exit");

    printf("\nEnter Your Choice : \n");
      scanf("%d",&ch);

      switch(ch)
      {
      case 1:

        printf("Enter The Push
Elements : ");
          scanf("%d",vl);

          Push(value,size);
           Display();

        goto menu;

      case 2:
            Pop();

        goto menu;

      case 3:
            Display();

        goto menu;


      case 4:
```

```
                printf("\n----Exit The
Program----");
                break;

            default:
                printf("\n--Please Enter The
Valid Choice....\n");

            }

            return 0;

}
```

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

#define MAX 50
char stack[MAX];
int top=-1;

int iS_operator(char ch)
{

   if(ch=='+' || ch=='-' || ch=='*' || ch=='/'
|| ch=='^')
   {
      return 1;
   }
   return 0;
}

int IS_operand(char ch)
{
   return (ch>='0' && ch<='9')||(ch>='A'
&& ch<='Z')||(ch>='a' && ch<='Z');
}

void revers()
{
   char str[MAX];

      printf("Enter The String : ");
         scanf("%s",str);

            for(int i=0; str[i]!='\0'; i++)
            {
               if(IS_operand(str[i]) ||
iS_operator(str[i]))
```

```
            {
                stack[++top]=str[i];
            }
        }

        for(int i=0 ;str[i]!='\0'; i++)
        {
            if(top!=-1)
            {
                str[i]=stack[top--];
            }
        }

            printf("\n--Reverse Stack  :%s
",str);
}

int main()
{
    revers();

    return 0;
}

/*
 * Q-13.c
 *
 *  Created on: 03-Aug-2024
 *      Author: root
 */

#include<stdio.h>
#include<string.h>

#define MAX 100
```

```
int top=-1;
char stack[MAX];

void Push(char value)
{
   if(top==MAX-1)
   {
      printf("\n----Stack Is Overflow----\n");
   }

   else
   {
      top=top+1;
      stack[top]=value;
   }

}

char Pop()
{
   if(top==-1)
   {
      printf("\n---Stack Is Under Flow ---");
      return '\0';
   }
   else
   {
      return stack[top--];
   }
}

int oprator(char value)
{
   printf(" %c ",value);
   switch(value)
```

```
    {
       case '+':
       case '-':
             return 1;
       case '*':
       case '/':
             return 2;

       default:
          return 3;

    }

}



char operand(char ch)
{

    if((ch>='0' && ch<='9') || (ch>='A' &&
ch<='Z') || (ch>='a' && ch<='z')){
       return 1;
    }
    else return 0;
}

char peek()
{
    return stack[top];
}
void Infix_to_Postfix(char infix[] , char
postfix[])
{
    int i,j=0;
```

```
char token;

top=-1;

   for( i=0 ; infix[i]!='\0'; i++)
   {
      token=infix[i];

         if(operand(token))
         {
            postfix[j++]=token;
         }
         else if(token=='(')
         {
            Push(token);
         }

         else if(token==')')
         {
            while(top!=-1 && peek()!='(')
            {
               postfix[j++]=Pop();
            }
            if(top!=-1 && peek()=='(')
            {
               pop();
            }
         }


         //Operator

         else
         {
```

```
                while(top!=-1 &&
stack[top]!='(' &&
oprator(peek())>=oprator(token))
                {
                   postfix[j++]=Pop();
                }
                Push(token);
            }

        }

           while(top!=-1)
           {
              if(stack[top]=='(')
              {
                 printf("\n MisMatched
ParanTheses ...");
              }

                 postfix[j++]=Pop();
           }

        postfix[j]='\0';
}

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

        printf("\n Enter Infix Expression ");
           scanf("%s",infix);

           Infix_to_Postfix(infix,postfix);
```

```
        printf("\n PostFix Expression
%s\n",postfix);

        return 0;

}



/*
 * Q-15.c
 *
 *  Created on: 12-Aug-2024
 *      Author: root
 */

#include<stdio.h>

#define MAX 100

int stack[MAX];
int top=-1;

void Push(int value);
int Pop();
int is_Operator(char ch);
void Postfix_Evaluation();

void Push(int value)
{
   if(top==MAX-1)
   {
      printf("\n---Stack Is Overflow---\n");
   }
```

```
    stack[++top]=value;
}

int Pop()

{
    if(top==-1)
    {
        printf("\n---Stack Is Empty---\n");
    }
    return stack[top--];
}

int is_Operator(char ch)
{
    return(ch=='+' || ch=='-' || ch=='*' ||
ch=='/' || ch=='%');
}

void Postfix_Evaluation()
{
    char str[50];

        printf("\n--Enter Your Postfix String--
\n");
            scanf("%s",str);

            for(int i=0 ; str[i]!='\0' ;i++)
            {
                if(is_Operator(str[i]))
                {
                    if(top<1)
                    {
```

```
                printf("\n--Error : Not
Enough Operand For %c Operation At
Position %d---\n",str[i],i);

                    return;
                }

                else
                {
                   int n1=Pop();
                   int n2=Pop();
                   int ans;

                      switch(str[i])
                      {
                      case '+':
                            ans=n2+n1;
                         break;

                      case '-':
                            ans=n2-n1;
                         break;

                      case '*':
                            ans=n2*n1;
                         break;

                      case '/':

                               if(n1==0)
                               {
                                  printf("\n--Error
Division by Zero--\n");
                                  return ;
                               }
```

```
                    ans=n2/n1;
                  break;

                default:
                    printf("\n--Error :
Unsupported Operator %c--\n",str[i]);

                    return;

                }
                Push(ans);
            }
        }
        else
        {
            int value;

                printf("\n--Enter Value Of %c
: ",str[i]);
                scanf("%d",&value);

                Push(value);
        }
    }

    if(top==0)
    {
        printf("\n--Final Output--
%d\n",Pop());
    }
    else
    {
        printf("\n--Error : Invalid Postfix
Expression--\n");
    }
```

```
}

int main()
{
   Postfix_Evaluation();

   return 0;
}


/*
 * Q-16.c
 *
 *  Created on: 12-Aug-2024
 *      Author: root
 */

#include<stdio.h>
#include<string.h>

char str1[100];
char str2[100];
char str3[200];

int top1=-1;
int top2=-1;
int top3=-1;

void Push(char ch)
{
   str3[++top3]=ch;

   printf("\n--%c Push Onto Stack--\n",ch);
```

```
}

char Pop1()
{
       if(top1==-1)
       {
          printf("\n---Stack Is Empty---\n");
       }
       return str1[top1--];

}

char Pop2()
{
       if(top2==-1)
       {
          printf("\n---Stack Is Empty---\n");
       }
       return str2[top2--];

}

int main()

{
    printf("\n--Enter Your First String--\n");
       scanf("%s",str1);

    printf("\n--Enter Your Second String--
\n");
       scanf("%s",str2);

       for(int i=0 ; str1[i]!='\0' ;i++)
       {
          top1++;
```

```
    }
    for(int i=0 ; str2[i]!='\0' ;i++)
    {
       top2++;
    }

       while(top1!=-1 || top2!=-1)
       {
          if(top1>=0)
          {
             char ch1=Pop1();

                   Push(ch1);
          }
          if(top2>=0)
          {
             char ch2=Pop2();

                   Push(ch2);
          }
       }

       str3[++top3]='\0';

       printf("\n--Your merged String : %s--
\n",str3);

    return 0;
}
```

ASSIGNMENT -2

1. Write a program to perform all operations on simple queue.

```
#include<stdio.h>

#define size 3

int SQ[size];
int f=0;
int r=0;

void input()
{
   if(r==size)
   {
      printf("\n overflow \n");
   }
   else
   {
      r++;
      if(r==1)
      {
         f=1;
      }

      printf("\n Enter Value : ");
      scanf("%d",&SQ[r]);
   }
}
```

```
void display()
{
   if(f==0)
   {
      printf("\n Underflow ");
   }
   else
   {
      printf("\n Element : ");
      for(int i=f;i<=r;i++)
      {
         printf(" %d ",SQ[i]);
      }
   }
}

void delete()
{
   if(f==0)
   {
      printf("\n Underflow \n");
   }

   if(r==f)
   {
         r=0;
         f=0;
   }

   else
   {
      printf("\n Delete Element : %d",SQ[f]);
      f++;
   }
```

```
}

int main()
{
   int choice;

       menu :
          printf("\n 1. Insertion :-");
          printf("\n 2. Display :-");
          printf("\n 3. Deletion :-");
          printf("\n 4. Exit :-");

          printf("\n Enter Your Choice : ");
          scanf("%d",&choice);

             switch(choice)
             {
             case 1:
                input();
                goto menu;

             case 2:
                display();
                goto menu;

             case 3:
                delete();

                goto menu;

             case 4:
                printf("\n Thank You ..");

                break;
             default:
```

```
                printf("\n Please Enter Valid
Choice : ");
            }

            return 0;
}
```

2. Write a program to implement circular queue

```
#include<stdio.h>
#define size 3

int CQ[size];
int f=0;
int r=0;

void insert();
void display();
void delete();

void insert()
{
   if((r==size && f==1) || (r==f-1))
   {
      printf("\n Stack is Full ");
   }
   else
   {

         r++;

      if(r==1)
      {
```

```
            f=1;
        }
        printf("\n Enter Value : ");
        scanf("%d",&CQ[r]);

    }

}

void display()
{
    if(f==0)
    {
        printf("\n underflow ");
    }

    else
    {
        printf("\n Elemets : ");
        for(int i=f;i<=r;i++)
        {
            printf(" %d",CQ[i]);
        }
    }
}

void delete()
{
    if(f==0)
    {
        printf("\n Underflow ");
    }
    else
    {
```

```
        printf("\n Delete Element :
%d",CQ[f]);
        f++;
        if(f==r)
    {
        f=0;
        r=0;
    }
    else if(f==size)
    {
        f=1;
    }
    else
    {

    }

}
}

int main()
{
    int choice;

    menu:
        printf("\n1. Insert");
        printf("\n2. Display");
        printf("\n3. Delete");
        printf("\n4. Exit");

        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
```

```
        case 1:
            insert();
            goto menu;

        case 2:
            display();

            goto menu;

        case 3:
            delete();

            goto menu;

        case 4:
            printf("\n Thank you ");
            break;

        default:
            printf("\n Enter Valid Choice : ");
            break;
        }

        return 0;
}
```

3. Write a program to implement queue which works as a stack

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

int deque[size];
```

```
int f = -1, r = -1;

void insertRear();
void deleteFront();
void deleteRear();
void display();

void insertRear()
{
   if (r==size-1)
   {
      printf("\nDeque is Full\n");
   } else
   {
      int value;
      printf("\nEnter Value: ");
      scanf("%d", &value);

      if (f == -1)
      {
         f = 0;
      }
      r++;
      deque[r] = value;
   }
}

void deleteFront()
{
   if (f == -1)
   {
      printf("\nDeque is Empty\n");
   } else
   {
```

```
        printf("\nDeleted Element from Front:
%d\n", deque[f]);
        f++;
        if (f == r)
        {
            // Queue is now empty
            f = -1;
            r = -1;
        }
    }
}

void deleteRear()
{
    if (f == -1)
    {
        printf("\nDeque is Empty\n");
    }
    else
    {
        printf("\nDeleted Element from Rear:
%d\n", deque[r]);
        r--;
        if (f == r)
        {
            // Queue is now empty
            f = -1;
            r = -1;
        }
    }
}

void display()
{
    if (f == -1)
```

```
    {
        printf("\nDeque is Empty\n");
    } else
    {
        printf("\nElements in Deque: ");

        for(int i=f ;i<r;i++)
        {
            printf(" %d ",deque[i]);
        }
        printf("\n");
    }
}

int main() {
    int choice;

    menu:
    printf("\n1. Insert at Rear");
    printf("\n2. Delete from Front");
    printf("\n3. Delete from Rear");
    printf("\n4. Display");
    printf("\n5. Exit");

    printf("\nEnter your choice: ");
    scanf("%d", &choice);

    switch (choice)
    {
        case 1:
            insertRear();
            goto menu;

        case 2:
            deleteFront();
```

```
            goto menu;

        case 3:
            deleteRear();
            goto menu;

        case 4:
            display();
            goto menu;

        case 5:
            printf("\nThank you\n");
            break;

        default:
            printf("\nEnter Valid Choice\n");
            goto menu;
    }

    return 0;
}
```

4. Implement Input restricted double ended queue

```
#include<stdio.h>
#define size 3

int r=0;
int f=0;
int DQ[size];
```

```
void Insert()
{
   printf("\n Insertion :-");
   if(r==size)
   {
      printf("\n Overflow :-");
   }
   else
   {
   r++;
   if(r==1)
   {
      f=1;
   }
      printf("\n Enter The Value : ");
      scanf("%d",&DQ[r]);
   }
}

void Display()
{
   if(f==0)
   {
   printf("\n Underflow :-");
   }
   else
   {
      printf("\n Elements : ");
      for(int i=f;i<=r;i++)
      {
      printf(" %d ",DQ[i]);
      }
   }
}
void Delete()
```

```
{

        printf("\n Your Delete Choice :-");
        printf("\n 1. Left Deletion :-");
        printf("\n 2. Right Deletion :-");

        int C;

           printf("\n Enter Your Delete Choice :
");
           scanf("%d",&C);

           switch(C)
           {
           case 1:
              if(f==0)
              {
                 printf("\n Underflow :-");
              }
              else
              {
                 if(f==r)
                 {
                    printf("\n Queue RESET :-");
                    f=0;
                    r=0;
                 }
                 else
                 {
                    printf("\n Deleted Element :
%d ",DQ[f]);
                    f++;
                 }
              }
                 break;
```

```
        case 2:
            if(f==0)
            {
                printf("\n Underflow :-");
            }
            else
            {
                if(f==r)
                {
                    f=0;
                    r=0;
                }
                else
                {
                    printf("\n Deleted Element :
%d ",DQ[r]);
                    r--;
                }
            }
                    break;
        }
}

int main()
{
   back:
       printf("\n Your Choice List :- \n ");
       printf("\n 1. Insert :-");
       printf("\n 2. Display :-");
       printf("\n 3. Delete :-");
       printf("\n 4. Exit :-");

       int Choice;

       printf("\n Enter Your Choice : ");
```

```
        scanf("%d",&Choice);

        switch(Choice)
        {
           case 1:
              Insert();
              goto back;
              break;
           case 2:
              Display();
              goto back;
              break;
           case 3:
              Delete();
              goto back;
              break;
           case 4:
              printf("\n Thank You ");
              break;
           default :
              printf("\n Case is not found
pls,Re-Enter The Case Number : ");
              goto back;
              break;
        }
     return 0;
}
```

5. Write a program to implement queue by reversing front and rear pointer

```
#include<stdio.h>

#define size 3

int f=size+1;
int r=size+1;
int RQ[size];

void input()
{
   if(r==1)
   {
      printf("\n overflow ");
   }
   else
   {
      r--;
      if(r==size)
      {
         f=size;
      }
      printf("\n Enter Value : ");
      scanf("%d",&RQ[r]);
   }
}

void diaplay()

{
   printf("\n Display Details : ");

   if(f==size+1)
   {
```

```
        printf("\n Underflow ");
    }
    else
    {
        printf("\n Elements : ");
        for(int i=r;i<=f;i++)
        {
            printf(" %d ",RQ[i]);
        }
    }
}

void delete()
{
    if(f==size+1)
    {
        printf("\n Underflow :");
    }
    else
    {
        if(f==r)
        {
            printf("\n Reset Queue : ");
            f=size;
            r=size;
        }
        else
        {
            printf("\n Delete Element : %d
",RQ[f]);
            f--;
        }
    }
}
```

```
int main()
{
   menu:
      printf("\n 1. insert :-");
      printf("\n 2. Display:-");
      printf("\n 3. Delete :-");
      printf("\n 4. Exit. :-");

      int choice;

         printf("\n Enter Your choice : ");
            scanf("%d",&choice);

         switch(choice)
         {
            case 1:
               input();

               goto menu;

            case 2:
               diaplay();

               goto menu;

            case 3:
               delete();

               goto menu;

            case 4:

               printf("\n Thank You ");

               break;
```

```
            default:
                printf("\n Please Enter Valid
Choice ");
                break;
        }

        return 0;
}
```

6. Write a program to implement a queue with integer pointer

```
#include<stdio.h>

#define size 3
int r=0;
int f=0;
int PQ[size];
int *p;

void Insert();
void Display();
void Delete();

void Insert()
{
   p=PQ;
      printf("\n Insertion :-");
   if(r==size)
   {
      printf("\n Overflow :-");
```

```
    }
    else
    {
        r++;
        if(r==1)
        {
        f=1;
        }
        printf("\n Enter The Value : ");
        scanf("%d",(p+r));
    }
}
void Display()
{
    p=PQ;
    if(f==0)
    {
        printf("\n Underflow :-");
    }
    else
    {
        for(int i=f;i<=r;i++)
    {
        printf("\n Element : %d ",*(p+i));
    }
    }
}
void Delete()
{
    p=PQ;
    if(f==0)
    {
        printf("\n Underflow :-");
    }
    else
```

```
    {
        if(f==r)
        {
            printf("\n Reset Queue :-");
            f=0;
            r=0;
        }
    else
    {
        printf("\n Deleted Element : %d
",*(p+f));
        f++;
    }
    }
}

int main()
{
    back:
        printf("\n Your Choice List :- \n ");
        printf("\n 1. Insert :-");
        printf("\n 2. Display :-");
        printf("\n 3. Delete :-");

        printf("\n 4. Exit :-");
        int Choice;

        printf("\n Enter Your Choice : ");
        scanf("%d",&Choice);

        switch(Choice)
        {
            case 1:

                Insert();
```

```
            goto back;
            break;

        case 2:

            Display();
            goto back;
            break;

        case 3:

            Delete();
            goto back;
            break;

        case 4:
            printf("\n Thank You ");
            break;

        default :

            printf("\n Case is not found
pls,Re-Enter The Case Number : ");
            goto back;

        break;
      }
   return 0;
}
```

7. Implement multiple priority queue with 2 priority queue as simple queue.

```
#include<stdio.h>

#define size 3

int f1=0;
int f2=0;

int r1=0;
int r2=0;

int SQ1[size];
int SQ2[size];

void insert()
{

	printf("\n Insertion :- ");

	printf("\n 1. Insert 1 :-");
	printf("\n 2. Insert 2 :-");

	int cho;
	   printf("\n Enetr Your Choice :- ");
	      scanf("%d",&cho);

	      switch(cho)
	      {
	         case 1:

	            if(r1==size)
	            {
	               printf("\n Overflow ");
```

```
    }
    else
    {
       r1++;
       if(r1==1)
       {
          f1=1;

       }

       printf("\n Enter Value : ");
          scanf("%d",&SQ1[r1]);
    }

       break;

case 2:

    if(r2==size)
    {
       printf("\n Overflow ");
    }
    else
    {
       r2++;
       if(r2==1)
       {
          f2=1;
       }
       printf("\n Enter value : ");
          scanf("%d",&SQ2[r2]);
    }

    break;
```

```
            }

}

void display()
{
   if(f1==0)
   {
      printf("\n Underflow 1");
   }

   else
   {
      printf("\n Element 1 :- ");

      for(int i=f1;i<=r1;i++)
      {
         printf(" %d ",SQ1[i]);
      }
   }
   if(f2==0)
   {
      printf("\n Underflow 2 ");
   }

   else
   {
      printf("\n Elemet 2 :- ");

      for(int i=f2;i<=r2;i++)
      {
         printf(" %d ",SQ2[i]);
      }
   }
}
```

```
void delete()
{

      printf("\n Delete 1 :-");
      printf("\n Delete 2 :-");

      int cho;
         printf("\n Enter value : ");
         scanf("%d",&cho);

      switch (cho)
      {

         case 1:

         if(f1==0)
         {
            printf("\n Underflow : ");
         }

            if(f1==r1)
            {
               printf("\n Reset Queue :-");
               f1=0;
               r1=0;

            }
            else
            {
               printf("\n Delete elemet :
%d",SQ1[f1]);
               f1++;

            }
```

```
            break;

        case 2:

        if(f1!=0)
        {
            printf("\n Sorry Queue 1 Is Full ");
        }
        else
        {
            if(f2==0)
            {
                printf("\n underflow");
            }
            if(f2==r2)
            {
                printf("\n Reset Queue :-");
                f2=0;
                r2=0;
            }
            else
            {
                printf("\n Deleted Element : %d
",SQ2[f2]);
                f2++;
            }
        }

        break;
    }

}

int main()
{
```

```
int choice;

menu:
   printf("\n1. Insert");
   printf("\n2. Display");
   printf("\n3. Delete");
   printf("\n4. Exit");

   printf("\nEnter your choice: ");
   scanf("%d", &choice);

   switch (choice)
   {
   case 1:
      insert();
      goto menu;

   case 2:
      display();

      goto menu;

   case 3:
      delete();

      goto menu;

   case 4:
      printf("\n Thank you ");
      break;

   default:
      printf("\n Enter Valid Choice : ");
      break;
   }
```

```
    return 0;
}
```

ASSIGNMENT – 3

1. Singly Linked list, with all operations.
(including sorting)

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

int x;

struct Node
{
   int data;
   struct Node *next;

}*first=NULL , *last=NULL , *nn=NULL ,
*cur , *pre , *temp;

void create()
{
   printf("\n Enter The value (-1 is Exit )");
   scanf("%d",&x);

      while (x != -1)
      {
         nn=(struct Node
*)malloc(sizeof(struct Node));
```

```
            nn->data=x;
            nn->next=NULL;

               if(first==NULL)
               {
                  first=nn;
                  last=nn;
               }
               else
               {
                  last->next=nn;
                  last=nn;

               }

               printf("\n Enter The value (-1 is
Exit )");
               scanf("%d",&x);
         }

}

void display()
{
   temp=first;

         while (temp != last)

         {
            printf("\n Elements : %d ",temp-
>data);
            temp=temp->next;
         }
         printf("\n Elements : %d ",temp-
>data);
```

```
}

void Insert_first()
{
   printf("\n Enter The Data : ");
   scanf("%d",&x);

      nn=(struct Node
*)malloc(sizeof(struct Node));

         nn->data=x;
         nn->next=first;
         first=nn;
}

void Insert_middel()
{
   int pos;
   int count=1;
   pre=NULL;
   cur=first;

      printf("\n Enter The Possition : ");
      scanf("%d",&pos);

         printf("\n Enter The Data : ");
         scanf("%d",&x);

      nn=(struct Node
*)malloc(sizeof(struct Node));

      while (count<pos)
      {
         pre=cur;
```

```
        cur=cur->next;
        count++;
      }

      nn->data=x;
      pre->next=nn;
      nn->next=cur;


}

void Insert_last()
{
   printf("\n Enter Data : ");
   scanf("%d",&x);

      nn=(struct Node
*)malloc(sizeof(struct Node));

        nn->data=x;
        last->next=nn;
        last=nn;
        nn->next=NULL;
}

void Delete_first()
{
   if(first==NULL)
   {
      printf("\n Under Flow ");

   }
   else
   {
      temp=first;
```

```
        first=first->next;
        free(temp);
    }

}

void Delete_middel()
{
    int pos;
    int count=1;
    pre=NULL;
    cur=first;

        printf("\n Enter The Possition : ");
        scanf("%d",&pos);

        while(count<pos);
        {
            pre=cur;
            cur=cur->next;
            count++;
        }

        pre->next=cur->next;
        free(cur);
}

void Delete_last()
{
    temp=first;

        while(temp->next!=last)
        {
            temp=temp->next;
        }
```

```
        free(temp);
        last=temp;
        last->next=NULL;
}

void shorting()
{
    pre=first;
    cur=first->next;

        while(cur!=NULL)
        {
            temp=pre;

            while(pre!=NULL)
            {
                if(temp->data > pre->data)
                {
                    int value = temp->data;
                    temp->data = pre->data;
                    pre->data = value;
                }

                pre = pre->next;
            }

            pre = cur;
            cur=cur->next;
        }
}

int main()
{
    create();
    int c;
```

```
menu:

   printf("\n 1. Insert first : ");
   printf("\n 2. Insert Middel : ");
   printf("\n 3. insert Last : ");
   printf("\n 4. Delete first : ");
   printf("\n 5. Delete Middel : ");
   printf("\n 6. Delete Last : ");
   printf("\n 7. Display");
   printf("\n 8. Shorting : ");

      printf("\n Enter your Choice : ");
      scanf("%d", &c);

   switch (c)
   {
   case 1:

         Insert_first();

      goto menu;

   case 2:

         Insert_middel();

      goto menu;

   case 3:

         Insert_last();

      goto menu;

   case 4:
```

```
            Delete_first();
        goto menu;
    case 5:
            Delete_middel();
        goto menu;
    case 6:
            Delete_last();
        goto menu;
    case 7:
            display();
        goto menu;
    case 8:
        shorting();

        goto menu;
    case 9:
        printf("\n Program Is Exit Thank
You : ");
        break;
```

```
        default:

            printf("\n Please Valid Choice : ");

            break;
        }

        return 0;
}
```

2. Circular Singly linked list, with all operations

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

int x;

struct Node
{
   int data;
   struct Node *Next;

} *first = NULL, *last = NULL, *nn = NULL,
*pre, *cur, *temp;

void create()
{
   printf("\n Enter The Data (-1 to End ) :
");
```

```
   scanf("%d", &x);

   while (x != -1)
   {
      nn = (struct Node
*)malloc(sizeof(struct Node));

      nn->data = x;
      nn->Next = NULL;

      if (first == NULL)
      {
         first = nn;
         last = nn;
      }
      else
      {
         nn->Next = first;
         last->Next = nn;
         last = nn;
      }

      printf("\n Enter The Data (-1 to End ) :
");
      scanf("%d", &x);
   }
}

void Display()
{
   temp = first;

   while (temp != last)
   {
```

```
        printf("\n Element : %d", temp-
>data);
        temp = temp->Next;
    }

    printf("\n Element : %d", temp->data);
}

void Insert_First()
{
    printf("\n Enter The Data : ");
    scanf("%d", &x);

    nn = (struct Node *)malloc(sizeof(struct
Node));

    nn->data = x;
    nn->Next = first;
    first=nn;
}

void Insert_middel()
{
    int pos;
    int count = 1;

    pre = NULL;
    cur = first;

    printf("\n Enter The Possition : ");
    scanf("%d", &pos);

    printf("\n Enter The Data : ");
    scanf("%d", &x);
```

```
    nn = (struct Node *)malloc(sizeof(struct
Node));

    while (count < pos)
    {
        pre = cur;
        cur = cur->Next;
        count++;
    }

    nn->data = x;
    pre->Next = nn;
    nn->Next = cur;
}

void Insert_last()
{
    printf("\n Enter The Data : ");
    scanf("%d", &x);

    nn = (struct Node *)malloc(sizeof(struct
Node));

    nn->data = x;
    last->Next = nn;
    last = nn;
    nn->Next = NULL;
}

void Delete_first()
{
    if (first == NULL)
    {
        printf("\n Under Flow ");
    }
```

```
    else
    {
        temp = first;
        first = first->Next;
        free(temp);
    }
}

void Delete_middel()
{
    int pos;
    int count = 1;

    pre = NULL;
    cur = first;

        printf("\n Enter The Possition : ");
        scanf("%d",&pos);

    while (count < pos)
    {
        pre = cur;
        cur = cur->Next;
        count++;
    }

    pre->Next = cur->Next;
    free(cur);
}

void Delete_last()
{
    temp = first;

    while (temp->Next != last)
```

```
    {
        temp = temp->Next;
    }
    free(last);
    last = temp;
    last->Next = first;
}

int main()
{
    create();
    int c;
menu:

    printf("\n 1. Insert first : ");
    printf("\n 2. Insert Middel : ");
    printf("\n 3. insert Last : ");
    printf("\n 4. Delete first : ");
    printf("\n 5. Delete Middel : ");
    printf("\n 6. Delete Last : ");
    printf("\n 7. Display");

    printf("\n Enter your Choice : ");
    scanf("%d", &c);

    switch (c)
    {
    case 1:

        Insert_First();

        goto menu;

    case 2:
```

```
    Insert_middel();
    goto menu;
case 3:
    Insert_last();
    goto menu;
case 4:
    Delete_first();
    goto menu;
case 5:
    Delete_middel();
    goto menu;
case 6:
    Delete_last();
    goto menu;
case 7:
    Display();
    goto menu;
case 8:
```

```
        printf("\n Program Is Exit Thank You :
");

        break;

    default:

        printf("\n Please Valid Choice : ");

        break;
    }

    return 0;
}
```

3. Implementation of queue with linked list

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

void Creation();
void Display();
void Deletion();

int x;

struct Node
{
    int data;
    struct Node *next;
```

```
} *first = NULL, *last = NULL, *nn = NULL,
*cur, *pre, *temp;

void Creation()
{
   printf("\n Enter The Data (-1 To end) : ");
   scanf("%d",&x);

      while(x!=-1)
      {
         nn=(struct Node
*)malloc(sizeof(struct Node));

            nn->data=x;
            nn->next=NULL;

            if(first==NULL)
            {
               first=nn;
               last=nn;
            }
            else
            {
               last->next=nn;
               last=nn;
            }

            printf("\n Enter The Data (-1 To
end) : ");
            scanf("%d",&x);

      }
}

void Display()
```

```
{
   temp=first;

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

      printf("\n Element : %d",temp->data);

}

void Deletion()
{
   if(first==NULL)
   {
      printf("\n Underflow");
   }
   else
   {
      temp=first;
      first=first->next;
      free(temp);
   }
}

int main()
{
   menu:

      printf("\n Your choice List .. ");

      printf("\n 1. Insertion/Creation ");
```

```
        printf("\n 2. Display ");
        printf("\n 3. Deletion ");
        printf("\n 4. Exit : ");

        int choice;

           printf("\n Enter Your Choice : ");
           scanf("%d",&choice);

        switch (choice)
        {
        case 1:

              Creation();

           goto menu;

        case 2:

              Display();

           goto menu;

        case 3:

              Deletion();

           goto menu;

        case 4:

           printf("\n Program is Exit Thank You
: ");

           break;
```

```
        default:

            printf("\n Please Enter The Valid
Choice ");
            goto menu;

            break;
        }

        return 0;
}
```

4. Implementation of stack with linked list

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

int x;
struct Node
{
    int data;
    struct Node *Next;
} *first = NULL, *last = NULL, *nn = NULL,
*cur, *pre, *temp;

void Creation()
{
    printf("\n Enter The Data (-1 to End) : ");
    scanf("%d",&x);

        while (x!=-1)
        {
```

```
        nn=(struct Node
*)malloc(sizeof(struct Node));

        nn->data=x;
        nn->Next=NULL;

        if(first==NULL)
        {
           first=nn;
           last=nn;
        }
        else
        {
           last->Next=nn;
           last=nn;
        }

        printf("\n Enter The Data (-1 to End)
: ");
        scanf("%d",&x);
      }

}

void Display()
{
   temp=first;

      while (temp!=last)
      {
         printf("\n Element : %d ",temp-
>data);
         temp=temp->Next;
      }
      printf("\n Element : %d ",temp->data);
```

```
}

void Deletion()
{
   temp=first;

      while (temp->Next!=last)
      {
         temp=temp->Next;
      }
      free(last);
      last=temp;
      last->Next=NULL;

}

int main()
{
   menu:

      printf("\n Your choice List .. ");

      printf("\n 1. Insertion/Creation ");
      printf("\n 2. Display ");
      printf("\n 3. Deletion ");
      printf("\n 4. Exit : ");

      int choice;

         printf("\n Enter Your Choice : ");
         scanf("%d",&choice);

      switch (choice)
      {
```

```
        case 1:
            Creation();
          goto menu;
        case 2:
            Display();
          goto menu;
        case 3:
            Deletion();
          goto menu;
        case 4:
          printf("\n Program is Exit Thank You
: ");
          break;
        default:
          printf("\n Please Enter The Valid
Choice ");
          goto menu;
          break;
        }
        return 0;
```

```
}
```

5. Doubly linked list, will all operations and display
Display from first to last
Display from last to first

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

void creation();
void Insert_first();
void Insert_middel();
void Insert_last();
void Delete_first();
void Delete_middel();
void Delete_last();
// void Display();
void Display_First_Last();
void Display_last_First();

int x;

struct Node
{
   int data;
   struct Node *right;
   struct Node *Left;

} *first = NULL, *last = NULL, *nn = NULL,
*cur, *pre, *temp;
```

```
void creation()
{
   printf("\n Enter The Data (-1 to end) : ");
   scanf("%d", &x);

   while (x != -1)
   {
      nn = (struct Node
*)malloc(sizeof(struct Node));

      nn->data = x;
      nn->right = NULL;
      nn->Left = NULL;

      if (first == NULL)
      {
         first = nn;
         last = nn;
      }
      else
      {
         last->right = nn;
         nn->Left = last;
         last = nn;
      }

      printf("\n Enter The Data (-1 to end) :
");
      scanf("%d", &x);
   }
}

void Display_First_Last()
{
```

```
    temp = first;

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

void Display_last_First()
{
    temp = last;

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

void Insert_first()
{
    printf("\n Enter The Data : ");
    scanf("%d", &x);

    nn = (struct Node *)malloc(sizeof(struct
Node));

    nn->data = x;
    nn->right = first;
    first->Left = nn;
```

```
    first = nn;
    nn->Left = NULL;
}

void Insert_last()
{
    printf("\n Enter The Data : ");
    scanf("%d", &x);

    nn = (struct Node *)malloc(sizeof(struct
Node));

    nn->data = x;
    nn->Left = last;
    last->right = nn;
    nn->right = NULL;
    last = nn;
}

void Insert_middel()
{
    int pos;
    int count = 1;

    pre = NULL;

    printf("\n Enter The Possition : ");
    scanf("%d", &pos);

    printf("\n Enter The Data : ");
    scanf("%d", &x);

    nn = (struct Node *)malloc(sizeof(struct
Node));
    nn->data = x;
```

```
    cur = first;

    while (count < pos)
    {
        cur = cur->right;
        count;
    }
    nn->Left = cur->Left;
    nn->right = cur;
    cur->Left->right = nn;
    cur->Left = nn;
}

void Delete_first()
{

    temp = first;

    first = first->right;
    free(temp);
    first->Left = NULL;
}

void Delete_last()
{
    last = last->Left;
    last->right = NULL;
}

void Delete_middel()
{
    int pos;
    int count = 1;

    printf("\n Enter The Possition : ");
```

```
    scanf("%d", &pos);
    cur=first;


    while (count < pos)
    {
        cur = cur->right;
        count++;
    }
    cur->Left->right = cur->right;
    cur->right->Left = cur->Left;
    free(cur);
}

int main()
{
    creation();
menu:

    printf("\n Your Chice List : \n");

    printf("\n 1. Insert Fitst : ");
    printf("\n 2. Insert Middel : ");
    printf("\n 3. Insert Last : ");
    printf("\n 4. Delete First : ");
    printf("\n 5. Delete Middel : ");
    printf("\n 6. Delete Last : ");
    printf("\n 7. Display First To Last : ");
    printf("\n 8. Display Last To First : ");

    int choice;

    printf("\n Enter Your choice :");
    scanf("%d", &choice);
```

```
switch (choice)
{
case 1:
   Insert_first();
   goto menu;
case 2:
   Insert_middel();
   goto menu;
case 3:
   Insert_last();
   goto menu;
case 4:
   Delete_first();
   goto menu;
case 5:
   Delete_middel();
   goto menu;
case 6:
   Delete_last();
```

```
        goto menu;

    case 7:

        Display_First_Last();

        goto menu;

    case 8:

        Display_last_First();

    case 9:
        printf("\n Program Is Exit >>");
        break;

    default:

        printf("\n Please Enter Valid Choice :
");
        break;

        goto menu;
    }

    return 0;
}
```

6. Linked list for student data
Rollno, name, sem, sub1marks,
sub2marks, sub3marks, total
Operations: insertion, display, delete by
rollno, display student node
with highest marks

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

int x;
void Insertion();
void Display();
void Display_Highest_Marks();

struct Student {
   int Roll, Sem, Sub1, Sub2, Sub3;
   float total;
   char name[50];
   struct Student *next;
} *first = NULL, *last = NULL, *nn = NULL,
*temp,*cur,*pre;

void Insertion() {

   printf("\n Enter The Value 1 to insert or -
1 to stop: ");
   scanf("%d", &x);

   while (x != -1) {
      nn = (struct Student
*)malloc(sizeof(struct Student));

      printf("\n\n Enter The Student Details
>>>>");
      printf("\n Enter Roll No : ");
      scanf("%d", &nn->Roll);

      printf("\n Enter Name : ");
```

```
        scanf("%s", nn->name);

        printf("\n Enter The Sem : ");
        scanf("%d", &nn->Sem);

        printf("\n Enter The Sub 1 Marks : ");
        scanf("%d", &nn->Sub1);

        printf("\n Enter The Sub 2 Marks : ");
        scanf("%d", &nn->Sub2);

        printf("\n Enter The Sub 3 Marks : ");
        scanf("%d", &nn->Sub3);

        // Calculate total marks
        nn->total = nn->Sub1 + nn->Sub2 +
nn->Sub3;

        nn->next = NULL;

        if (first == NULL) {
            first = nn;
            last = nn;
        } else {
            last->next = nn;
            last = nn;
        }

        printf("\n Enter The Value 1 to insert
or -1 to stop: ");
        scanf("%d", &x);
    }
}

void Display() {
```

```
    temp = first;
    printf("\nRoll No\t Name \t Sem \t Sub1
\t Sub2 \t Sub3 \t Total");

    while (temp != NULL) {
        printf("\n%d\t %s \t %d \t %d \t %d \t
%d \t %.2f", temp->Roll, temp->name,
temp->Sem, temp->Sub1, temp->Sub2,
temp->Sub3, temp->total);
        temp = temp->next;
    }
}

void Delete_by_roll_no()
{
    if(first == NULL)
    {
        printf("\n\n Student Data is Empty ");
    }
    int delete;

        printf("\n Enter The Roll NO To delete
: ");
        scanf("%d",&delete);

    cur=first;
    pre=NULL;

        while(cur != NULL)
        {
            if(cur->Roll == delete)
            {
                printf("\nRoll No\t Name \t Sem
\t Sub1 \t Sub2 \t Sub3 \t Total");
```

```
            printf("\n%d\t %s \t %d \t %d \t
%d \t %d \t %.2f", cur->Roll, cur->name,
cur->Sem, cur->Sub1, cur->Sub2, cur-
>Sub3, cur->total);


            pre->next =cur->next;
            cur->next = NULL;
            free(cur);

         }

         pre=cur;
         cur=cur->next;
      }

      if(cur == NULL)
      {
         printf("\n\nRoll Number Is Not
Available.....");

      }

      if(cur == last)
      {
            printf("\nRoll No\t Name \t Sem
\t Sub1 \t Sub2 \t Sub3 \t Total");
            printf("\n%d\t %s \t %d \t %d \t
%d \t %d \t %.2f", cur->Roll, cur->name,
cur->Sem, cur->Sub1, cur->Sub2, cur-
>Sub3, cur->total);

         last=pre;
      }
```

```
        free(cur);


}


void Display_Highest_Marks()
{
    if (first == NULL)
    {
        printf("\n No students available.");
        return;
    }

    struct Student *max = first;
    temp = first->next;

    while (temp != NULL)
    {
        if (temp->total > max->total)
        {
            max = temp;
        }
        temp = temp->next;
    }

    printf("\nStudent with the highest
marks:");

    printf("\nRoll No: %d\nName: %s\nSem:
%d\nSub1: %d\nSub2: %d\nSub3:
%d\nTotal: %.2f",
           max->Roll, max->name, max->Sem,
max->Sub1, max->Sub2, max->Sub3, max-
>total);
```

```
}

int main() {
   int choice;


   menu:

      printf("\n\n Your Choice >>> ");
      printf("\n 1. Insertion ");
      printf("\n 2. Display ");
      printf("\n 3. Display Highrst Marks : ");
      printf("\n 4. Delete By Roll No ");
      printf("\n 5. Exit ");
      printf("\n Enter Your Choice : ");
      scanf("%d", &choice);

      switch (choice)
      {
         case 1:

            Insertion();

            goto menu;

         case 2:
            Display();

            goto menu;

         case 3:

            Display_Highest_Marks();

            goto menu;
```

```
        case 4:

            Delete_by_roll_no();

            goto menu;

        default:
            printf("\nInvalid Choice!");
            break;
    }

    return 0;
}
```

7. Merging of two linked list
Insert 2 values inside a node, character and integer, divide this list into two linked list into integer LL and character LL
Make addition of interger values of all the nodes in a SLL

```
#include <string.h>

char ch[100]; // Array to hold character input
int x;

// Structure for character node
struct CharNode {
    char character;
    struct CharNode* next;
```

```
} *first1 = NULL, *last1 =
NULL,*nn1=NULL,*temp1;

// Structure for integer node
struct IntNode {
    int integer;
    struct IntNode* next;
} *first2 = NULL, *last2 =
NULL,*nn2=NULL,*temp2;

// Structure for merged node containing
both integer and character
struct MergedNode {
    char character;
    int integer;
    struct MergedNode* next;
} *first3 = NULL, *last3 =
NULL,*nn3=NULL,*temp3;

// Function prototypes
void Insert();
void Display_Char_List();
void Display_Int_List();
void Merge_Lists();
void Display_Merged_List();

// Function to create and insert a node in
the main list
void Insert() {
    printf("\nEnter Character Data (-1 to
End): ");
    scanf("%s", ch);

    printf("Enter Integer Data (-1 to End): ");
    scanf("%d", &x);
```

```
   while (x != -1) {
      nn1 = (struct
CharNode*)malloc(sizeof(struct
CharNode));
      nn2 = (struct
IntNode*)malloc(sizeof(struct IntNode));

      nn1->character = ch[0]; // Assuming
you want the first character of the string
      nn1->next = NULL;
      nn2->integer = x;
      nn2->next = NULL;

      if (first1 == NULL) {
         first1 = nn1;
         last1 = nn1;
         first2 = nn2;
         last2 = nn2;
      } else {
         last1->next = nn1;
         last1 = nn1;
         last2->next = nn2;
         last2 = nn2;
      }

      printf("\nEnter Character Data (-1 to
End): ");
      scanf("%s", ch);

      printf("Enter Integer Data (-1 to End):
");
      scanf("%d", &x);
   }
}
```

```
void Display_Char_List()
{
   temp1 = first1;
   printf("\nCharacter List:");
   while (temp1 != NULL)
   {
      printf(" %c ->", temp1->character);
      temp1 = temp1->next;
   }
   printf(" NULL\n");
}

void Display_Int_List() {

   temp2 = first2;

   printf("\nInteger List:");
   while (temp2 != NULL)
   {
      printf(" %d ->", temp2->integer);
      temp2 = temp2->next;
   }
   printf(" NULL\n");
}

void Merge_Lists() {

   temp1 = first1;
   temp2 = first2;

   while (temp1 != NULL && temp2 !=
NULL)
   {
```

```
        nn3 = (struct
MergedNode*)malloc(sizeof(struct
MergedNode));

        nn3->character = temp1->character;
        nn3->integer = temp2->integer;
        nn3->next = NULL;

        if (first3 == NULL)
        {
            first3 = nn3;
            last3 = nn3;
        } else
        {
            last3->next = nn3;
            last3 = nn3;
        }

        temp1 = temp1->next;
        temp2 = temp2->next;
    }
}

void Display_Merged_List() {

    temp3 = first3;
    printf("\nMerged List:");
    while (temp3 != NULL)
    {
        printf(" (%c, %d) ->", temp3-
>character, temp3->integer);
        temp3 = temp3->next;
    }
    printf(" NULL\n");
}
```

```
// Main function
int main() {
    int choice;

    while (1) {
        printf("\n\nMenu:");
        printf("\n1. Insert (Character,
Integer)");
        printf("\n2. Display Character List");
        printf("\n3. Display Integer List");
        printf("\n4. Merge Character and
Integer Lists");
        printf("\n5. Display Merged List");
        printf("\n6. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                Insert();
                break;
            case 2:
                Display_Char_List();
                break;
            case 3:
                Display_Int_List();
                break;
            case 4:
                Merge_Lists();
                printf("Lists merged.\n");
                break;
            case 5:
                Display_Merged_List();
                break;
```

```
            case 6:
                exit(0);
            default:
                printf("Invalid choice!\n");
        }
    }

    return 0;
}
```

8. Create a Singly linked list to represent polynomial
9. Create a Singly linked list and perform sum all node values.

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

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

void Creation();
void Display();
void Addition();

int x;

struct Node
{
    int data;
```

```
    struct Node *next;
} *first = NULL, *last = NULL, *nn = NULL,
*cur, *pre, *temp;

void Creation()
{
    printf("\n Enter The Data (-1 To end) : ");
    scanf("%d",&x);

        while(x!=-1)
        {
            nn=(struct Node
*)malloc(sizeof(struct Node));

                nn->data=x;
                nn->next=NULL;

                if(first==NULL)
                {
                    first=nn;
                    last=nn;
                }
                else
                {
                    last->next=nn;
                    last=nn;
                }

                printf("\n Enter The Data (-1 To
end) : ");
                scanf("%d",&x);

        }
}
```

```
void Display()
{
   temp=first;

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

       printf("\n Element : %d",temp->data);

}

void Addition()
{
  int sum=0;

  temp=first;

       while (temp!=NULL)
       {
          sum=sum+temp->data;
          temp=temp->next;
       }
       printf("\n Addition Of Singly Linked
List : %d ",sum);

}

int main()
{
   menu:
```

```
printf("\n Your choice List .. ");

printf("\n 1. Insertion/Creation ");
printf("\n 2. Display ");
printf("\n 3. Addition ");
printf("\n 4. Exit : ");

int choice;

   printf("\n Enter Your Choice : ");
   scanf("%d",&choice);

switch (choice)
{
case 1:

      Creation();

   goto menu;

case 2:

      Display();

   goto menu;

case 3:

      Addition();

   goto menu;

case 4:
```

```
            printf("\n Program is Exit Thank You
: ");

            break;

        default:

            printf("\n Please Enter The Valid
Choice ");
            goto menu;

            break;
        }

        return 0;
}
```