(परीक्षार्थी द्वारा भरा जाए)

(To be filled by the Candidate)

July-December, 2020

परीक्षा का नाम (Name of Examination). **B.Tech. III Semester Examination, July-December, 2020**

अनुक्रमांक अंकों में (In figures) ......**1913075**..

अनुक्रमांक (शब्दों में) (Roll No. in Words) .... **nineteen lakh thirteen thousand seventy-five**..........

नामांकन संख्या (Enrollment No.) ....**2019/888**...

ई-मेल आई.डी. (E-mail ID.) .... **btbtc19047_riyanshi@banasthali.in**.........

विषय (Subject)...........................**Computer Science**.......................................

प्रश्नपत्र कोड सहित (Paper with Code) .... **CS-209 Data Structure Lab Record……………..**

Total Number of Pages: __113_______

Signature of the Student

Riyanshi

Riyanshi

1913075

INDEX

Riyanshi

Riyanshi

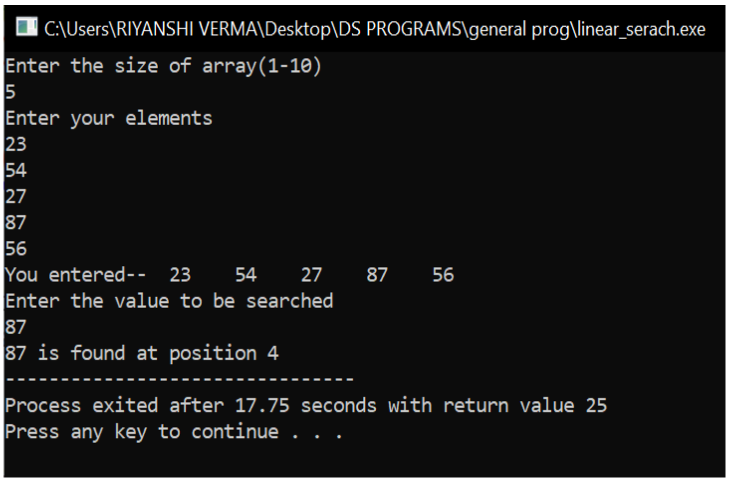
1913075

LINEAR SEARCH

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
       int n;
       printf("Enter the size of array(1-10)\n");
       scanf("%d",&n);
       if(n<0 || n>10)
       {
              printf("Enter correct value of n\n ");
              scanf("%d",&n);
       }
      if(n<0 || n>10)
      {
              exit(0);
      }
      int arr[10];
      int i;
      printf("Enter your elements\n");
      for(i=0;i<n;i++)
      {
             scanf("%d",&arr[i]);
      }
      printf("You entered--  ");
      for(i=0;i<n;i++)
      {
             printf("%d    ",arr[i]);
      }
      printf("\nEnter the value to be searched\n");
      int value;
      scanf("%d",&value);
      for(i=0;i<n;i++)
      {
             if(arr[i]==value)
             break;
      }
      if(i<n)
             printf("%d is found at position %d",value,i+1);
      else
             printf("%d is not found in the given array",value);
}
```

Riyanshi

Riyanshi



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\linear_serach.exe
Enter the size of array(1-10)
5
Enter your elements
23
54
27
87
56
You entered--  23    54    27    87    56
Enter the value to be searched
87
87 is found at position 4
--------------------------------
Process exited after 17.75 seconds with return value 25
Press any key to continue . . .
```

****

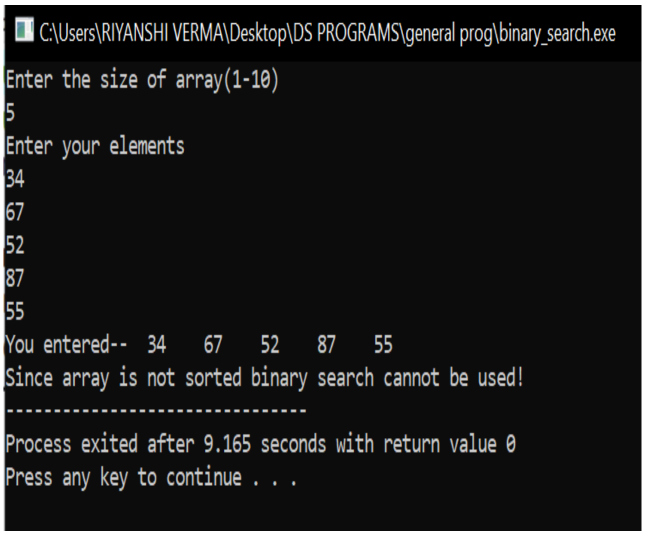
BINARY SEARCH

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
	int n;
	printf("Enter the size of array(1-10)\n");
	scanf("%d",&n);
	if(n<0 || n>10)
	{
		printf("Enter correct value of n\n ");
		scanf("%d",&n);
	}
	if(n<0 || n>10)
	{
		exit(0);
	}
	int arr[10];
	int i;
	printf("Enter your elements\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&arr[i]);
	}
	printf("You entered--  ");
	for(i=0;i<n;i++)
	{
		printf("%d    ",arr[i]);
	}

	for(i=0;i<n-1;i++)
	{
	        if(arr[i]<arr[i+1])
```

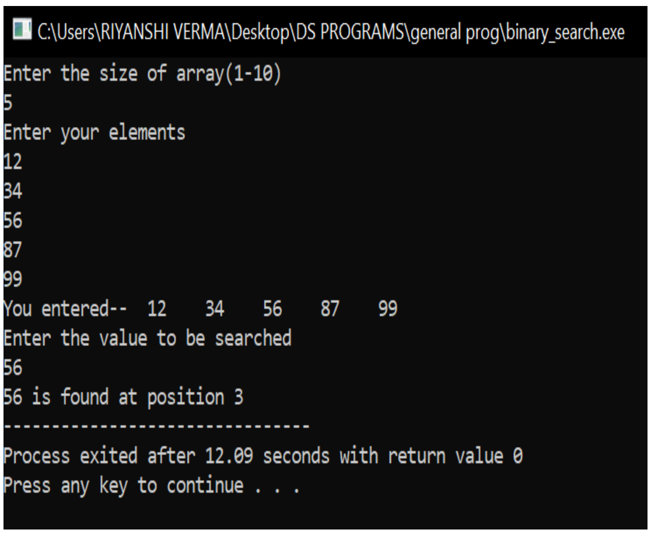
Riyanshi

Riyanshi

```
                continue;
                else
                break;
        }
        if(i!=n-1)
        {
                printf("\nSince array is not sorted binary search cannot be used!");
                exit(0);
        }
        printf("\nEnter the value to be searched\n");
        int value;
        scanf("%d",&value);
        int left=0,right=n-1,mid;
        while(left<=right)
        {
                mid=(left+right)/2;
                if(arr[mid]<value)
                left=mid+1;
                else if(arr[mid]==value)
                {
                        printf("%d is found at position %d",value,mid+1);break;
                }
                else
                right=mid-1;
        }
        if(left>right)
        printf("%d is not found in the given array\n",value);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\binary_search.exe
Enter the size of array(1-10)
5
Enter your elements
34
67
52
87
55
You entered--  34    67    52    87    55
Since array is not sorted binary search cannot be used!
------------------------------
Process exited after 9.165 seconds with return value 0
Press any key to continue . . .
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\binary_search.exe
Enter the size of array(1-10)
5
Enter your elements
12
34
56
87
99
You entered--  12    34    56    87    99
Enter the value to be searched
56
56 is found at position 3
------------------------------
Process exited after 12.09 seconds with return value 0
Press any key to continue . . .
```

****

<u>BUBBLE SORT</u>

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

Riyanshi

Riyanshi

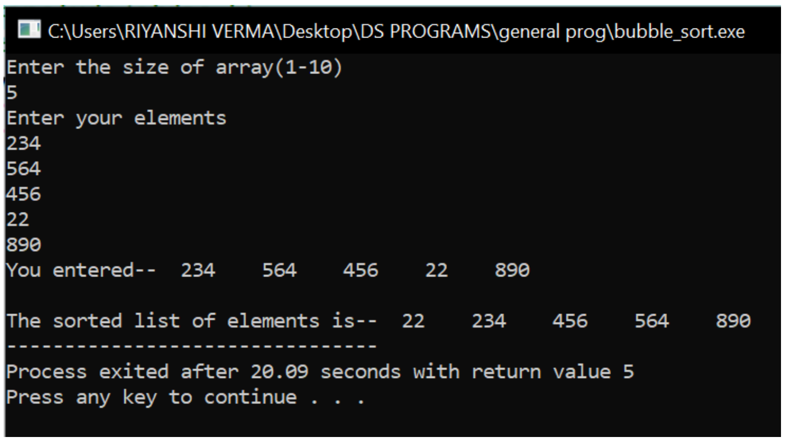
1913075

```
{
        int n;
        printf("Enter the size of array(1-10)\n");
        scanf("%d",&n);
        if(n<0 || n>10)
        {
                printf("Enter correct value of n\n ");
                scanf("%d",&n);
        }
        if(n<0 || n>10)
        {
                exit(0);
        }
        int arr[10];
        int i;
        printf("Enter your elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("You entered--  ");
        for(i=0;i<n;i++)
        {
                printf("%d    ",arr[i]);
        }
        int j,temp;
        for(i=0;i<n-1;i++)
        {
                for(j=0;j<(n-i-1);j++)
                {
                        if(arr[j]>arr[j+1])
                        {
                                temp=arr[j];
                                arr[j]=arr[j+1];
                                arr[j+1]=temp;
                        }
                }
        }
        printf("\n\nThe sorted list of elements is--  ");
        for(i=0;i<n;i++)
        {
                printf("%d    ",arr[i]);
        }

}
```

Riyanshi

Riyanshi

1913075



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\bubble_sort.exe
Enter the size of array(1-10)
5
Enter your elements
234
564
456
22
890
You entered--  234    564    456    22    890

The sorted list of elements is--  22    234    456    564    890
--------------------------------
Process exited after 20.09 seconds with return value 5
Press any key to continue . . .
```

****

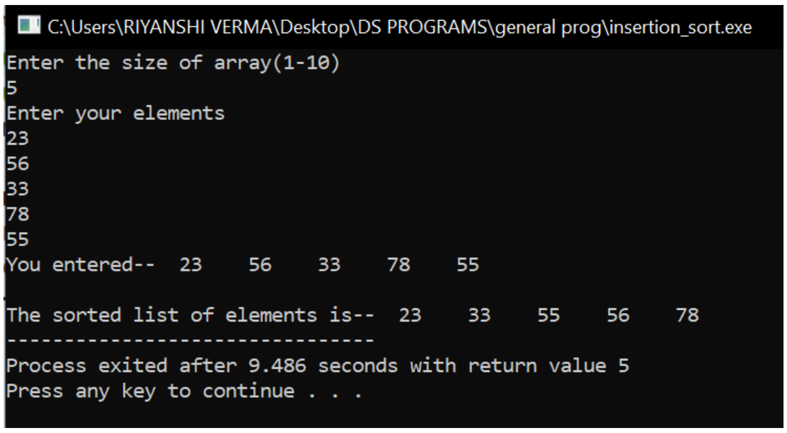
INSERTION SORT

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
        int n;
        printf("Enter the size of array(1-10)\n");
        scanf("%d",&n);
        if(n<0 || n>10)
        {
                printf("Enter correct value of n\n ");
                scanf("%d",&n);
        }
        if(n<0 || n>10)
        {
                exit(0);
        }
        int arr[10];
        int i;
        printf("Enter your elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("You entered--  ");
        for(i=0;i<n;i++)
        {
                printf("%d    ",arr[i]);
        }
        int j,x;
        for(i=1;i<n;i++)
        {
                j=i;x=arr[i];
                while(arr[j-1]>x && j>0)
                {
                        arr[j]=arr[j-1];j=j-1;
                }
```

Riyanshi

Riyanshi

```
			if (j!=i)
			arr[j]=x;
		}
		printf("\n\nThe sorted list of elements is--  ");
		for(i=0;i<n;i++)
		{
			printf("%d    ",arr[i]);
		}

}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\insertion_sort.exe
Enter the size of array(1-10)
5
Enter your elements
23
56
33
78
55
You entered--  23    56    33    78    55

The sorted list of elements is--  23    33    55    56    78
--------------------------------
Process exited after 9.486 seconds with return value 5
Press any key to continue . . .
```

****

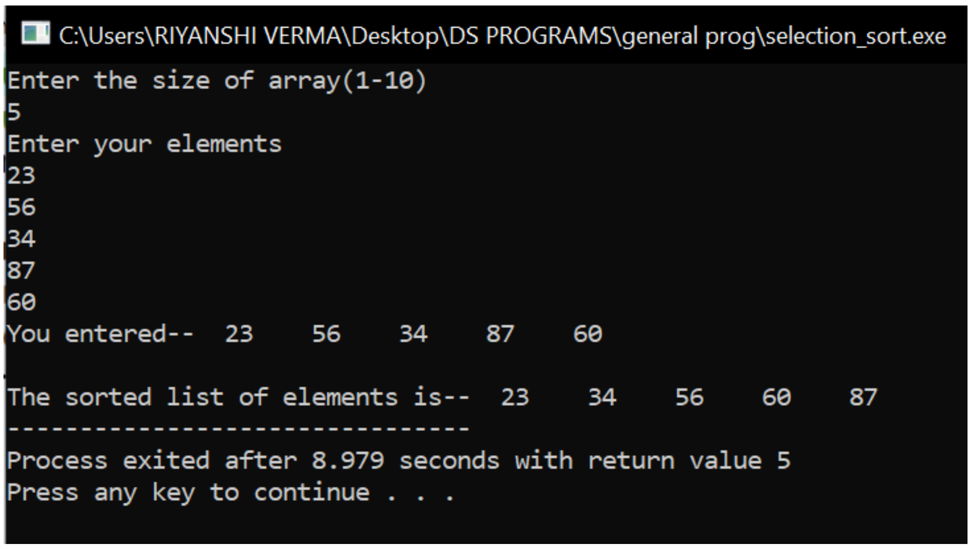
SELECTION SORT

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
	int n;
	printf("Enter the size of array(1-10)\n");
	scanf("%d",&n);
	if(n<0 || n>10)
	{
		printf("Enter correct value of n\n ");
		scanf("%d",&n);
	}
	if(n<0 || n>10)
	{
		exit(0);
	}
	int arr[10];
	int i;
	printf("Enter your elements\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&arr[i]);
	}
	printf("You entered--  ");
```

Riyanshi

Riyanshi

```
	for(i=0;i<n;i++)
	{
		printf("%d    ",arr[i]);
	}
	int j,temp,min;
	for(i=0;i<n-1;i++)
	{
	   min=i;
		for(j=i+1;j<n;j++)
		{
			if(arr[j]>arr[min])
			{
				min=j;
			}
			temp=arr[j];
		   arr[j]=arr[min];
		   arr[min]=temp;
		}
	}
	printf("\n\nThe sorted list of elements is--  ");
	for(i=0;i<n;i++)
		printf("%d    ",arr[i]);

}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\selection_sort.exe
Enter the size of array(1-10)
5
Enter your elements
23
56
34
87
60
You entered--  23    56    34    87    60

The sorted list of elements is--  23    34    56    60    87
--------------------------------
Process exited after 8.979 seconds with return value 5
Press any key to continue . . .
```

****

INSERTION AND DELETION DIFFERENT CASES

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
	int n;
	printf("Enter the size of array(1-10)\n");
```

Riyanshi

Riyanshi

1913075

```
        scanf("%d",&n);
        if(n<0 || n>10)
        {
                printf("Enter correct value of n\n ");
                scanf("%d",&n);
        }
        if(n<0 || n>10)
        {
                exit(0);
        }
        int arr[100];
        int i;
        printf("Enter your elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("You entered--  ");
        for(i=0;i<n;i++)
        {
                printf("%d    ",arr[i]);
        }
        int ch=0;
        do
        {     if(n<1)
           {
                    printf("\n\ndeletion might not be possible choose only INSERTION operations");

        }
        if(n>9)
            {
                    printf("\n\ninsertion might not be possible choose only DELETION operations");

        }
        printf("\n\n\nmenu driven program\n1. To insert element at specific position\n2. To insert element at
beginning\n3. To insert element at end\n4. To insert an element before a specified index\n5. To insert an
element after the given index\n6. To delete element at specific position\n7. To delete element at
beginning\n8. To delete element at end\n9. To delete an element before a specified index\n10. To delete an
element after the given index\n11. To reverse the array \n12. To end program");
            printf("\n\nEnter you choice\n");
            scanf("%d",&ch);
                    if(n<1 && ch>5)
                    {
                            printf("DELETION NOT POSSIBLE\nTHANKS.....");exit(0);
                    }

                    if(n>9 && ch<6)
                    {
                            printf("INSERTION NOT POSSIBLE\nTHANKS.....");exit(0);
                    }

                    int pos,val;int temp;
```

Riyanshi

Riyanshi

1913075

```
switch(ch)
{
            case 1:  printf("Enter the position\n");
                        scanf("%d",&pos);
                        printf("Enter the value to be inserted\n");
                        scanf("%d",&val);
                        n=n+1;
                       for (i = n; i >= pos; i--)
                       arr[i] = arr[i - 1];
                       arr[pos - 1] = val;
                       printf("The required result is--   ");
                       for (i = 0; i < n; i++)
                      printf("%d    ", arr[i]);
                      break;

          case 2: pos=1;
                     printf("Enter the value to be inserted\n");
                     scanf("%d",&val);
                     n=n+1;
                     for (i = n; i >= pos; i--)
                    arr[i] = arr[i - 1];
                    arr[pos - 1] = val;
                    printf("The required result is--   ");
                    for (i = 0; i < n; i++)
                    printf("%d   ", arr[i]);
                    break;

        case 3: pos=n;
                   printf("Enter the value to be inserted\n");
                   scanf("%d",&val);
                   n=n+1;
                   arr[n-1] = val;
                   printf("The required result is--   ");
                   for (i = 0; i < n; i++)
                   printf("%d   ", arr[i]);
                   break;

        case 4: printf("Enter the position\n");
                   scanf("%d",&pos);
                   pos=pos-1;
                   printf("Enter the value to be inserted\n");
                   scanf("%d",&val);
                   n=n+1;
                   for (i = n; i >= pos; i--)
                   arr[i] = arr[i - 1];
                   arr[pos - 1] = val;
                  printf("The required result is--   ");
                  for (i = 0; i < n; i++)
                  printf("%d   ", arr[i]);
                  break;

       case 5: printf("Enter the position\n");
```

Riyanshi

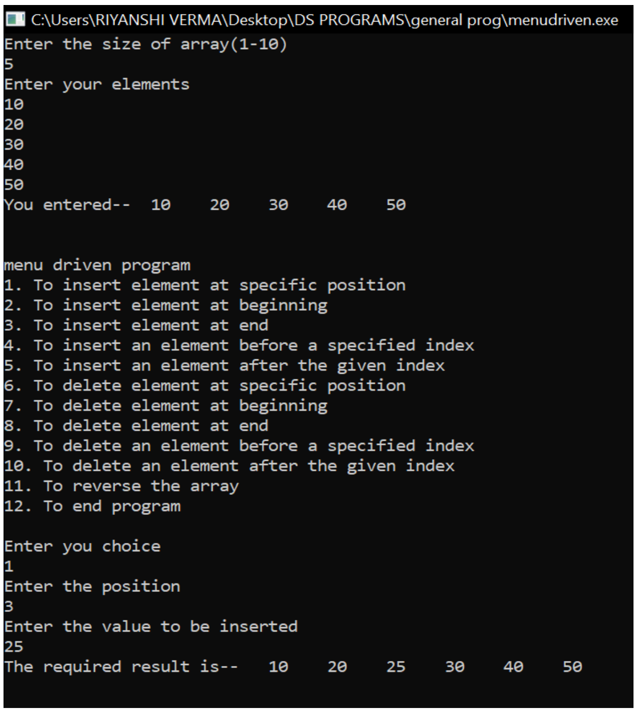
Riyanshi

```
              scanf("%d",&pos);
              pos=pos+1;
              printf("Enter the value to be inserted\n");
              scanf("%d",&val);
              n=n+1;
              for (i = n; i >= pos; i--)
              arr[i] = arr[i - 1];
             arr[pos - 1] = val;
             printf("The required result is--   ");
             for (i = 0; i < n; i++)
             printf("%d   ", arr[i]);
             break;

 case 6: printf("Enter the position\n");
              scanf("%d",&pos);
              n=n-1;
              for (i = pos-1; i <n; i++)
              arr[i] = arr[i + 1];
              printf("The required result is--   ");
              for (i = 0; i < n; i++)
              printf("%d    ", arr[i]);
              break;

  case 7: pos=1;
              n=n-1;
              for (i = pos-1; i <n; i++)
              arr[i] = arr[i + 1];
              printf("The required result is--   ");
              for (i = 0; i < n; i++)
              printf("%d   ", arr[i]);
             break;

 case 8: pos=n;
              n=n-1;
              printf("The required result is--   ");
              for (i = 0; i < n; i++)
              printf("%d   ", arr[i]);
              break;

  case 9: printf("Enter the position\n");
              scanf("%d",&pos);
              pos=pos-1;
              n=n-1;
              for (i = pos-1; i <n; i++)
             arr[i] = arr[i + 1];
             printf("The required result is--   ");
             for (i = 0; i < n; i++)
             printf("%d   ", arr[i]);
             break;
case 10: printf("Enter the position\n");
              scanf("%d",&pos);
              pos=pos+1;
```

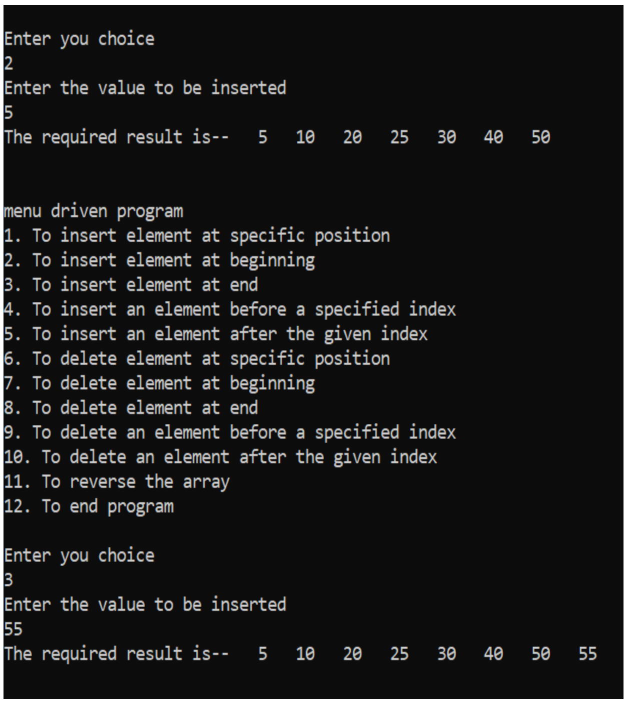
Riyanshi

Riyanshi

```
                        n=n-1;
                        for (i = pos-1; i <n; i++)
                        arr[i] = arr[i + 1];
                        printf("The required result is--   ");
                        for (i = 0; i < n; i++)
                        printf("%d   ", arr[i]);
                        break;
                case 11:
                        for(i=0;i<n/2;i++)
                        {
                                temp=arr[i];
                                arr[i]=arr[n-1-i];
                                arr[n-1-i]=temp;
                         }
                        printf("\n\nreversed array--  ");
                        for(i=0;i<n;i++)
                        {
                                printf("%d    ",arr[i]);
                        }
                        break;
               case 12: printf("\nTHANKS.....");
                        exit(0);
                        break;
              }
      }while(ch!=12);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\menudriven.exe
Enter the size of array(1-10)
5
Enter your elements
10
20
30
40
50
You entered--  10    20    30    40    50


menu driven program
1. To insert element at specific position
2. To insert element at beginning
3. To insert element at end
4. To insert an element before a specified index
5. To insert an element after the given index
6. To delete element at specific position
7. To delete element at beginning
8. To delete element at end
9. To delete an element before a specified index
10. To delete an element after the given index
11. To reverse the array
12. To end program

Enter you choice
1
Enter the position
3
Enter the value to be inserted
25
The required result is--   10    20    25    30    40    50
```

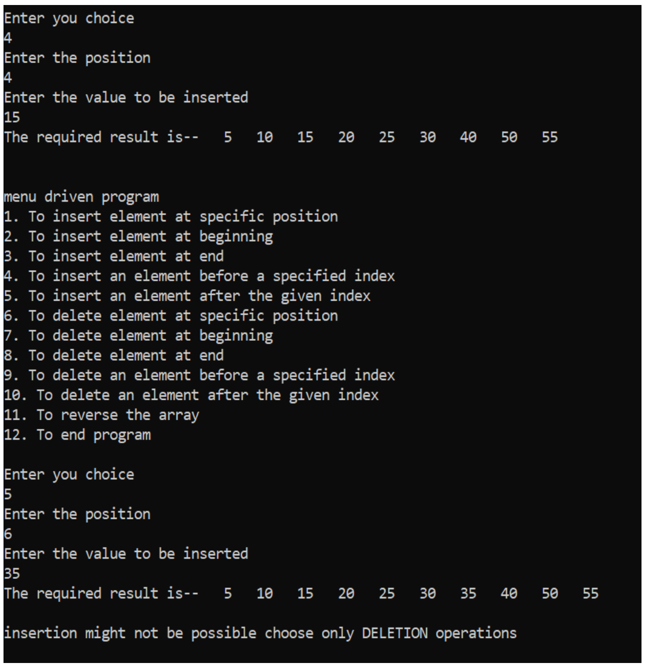


```
Enter you choice
2
Enter the value to be inserted
5
The required result is--   5   10   20   25   30   40   50


menu driven program
1. To insert element at specific position
2. To insert element at beginning
3. To insert element at end
4. To insert an element before a specified index
5. To insert an element after the given index
6. To delete element at specific position
7. To delete element at beginning
8. To delete element at end
9. To delete an element before a specified index
10. To delete an element after the given index
11. To reverse the array
12. To end program

Enter you choice
3
Enter the value to be inserted
55
The required result is--   5   10   20   25   30   40   50   55
```

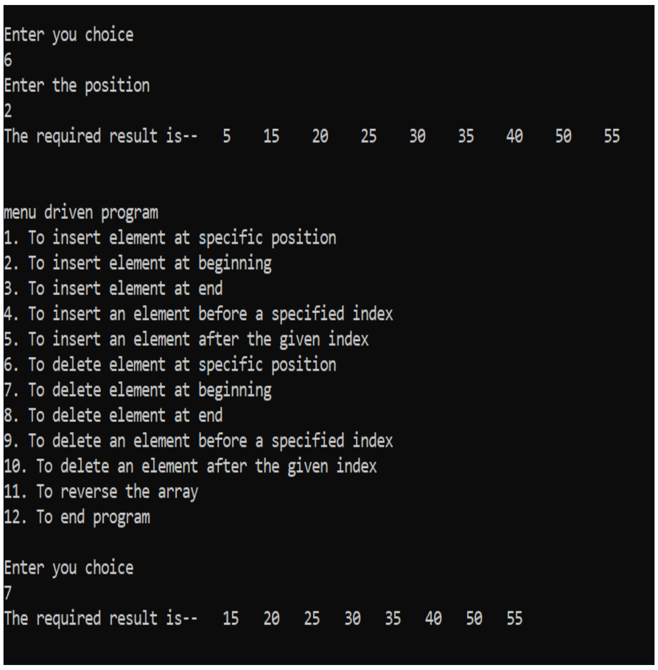
Riyanshi

Riyanshi

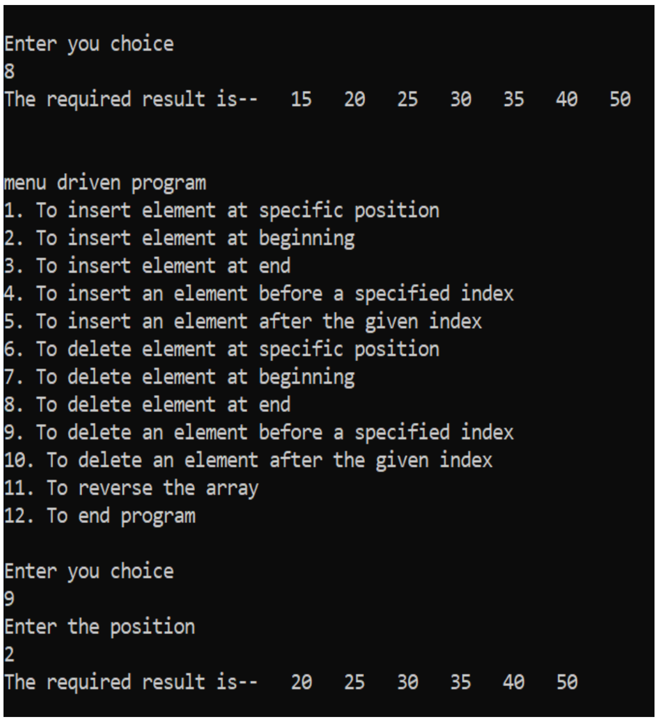
1913075



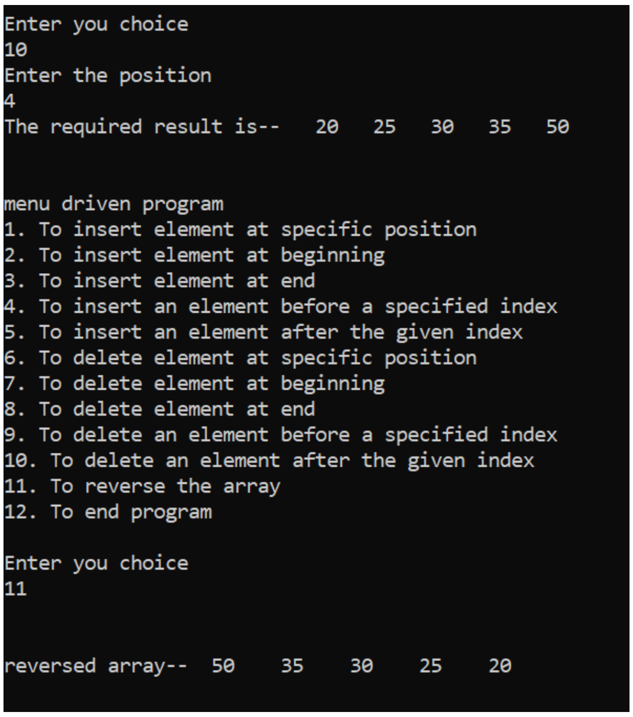
```
Enter you choice
4
Enter the position
4
Enter the value to be inserted
15
The required result is--   5   10   15   20   25   30   40   50   55


menu driven program
1. To insert element at specific position
2. To insert element at beginning
3. To insert element at end
4. To insert an element before a specified index
5. To insert an element after the given index
6. To delete element at specific position
7. To delete element at beginning
8. To delete element at end
9. To delete an element before a specified index
10. To delete an element after the given index
11. To reverse the array
12. To end program

Enter you choice
5
Enter the position
6
Enter the value to be inserted
35
The required result is--   5   10   15   20   25   30   35   40   50   55

insertion might not be possible choose only DELETION operations
```



```
Enter you choice
6
Enter the position
2
The required result is--   5   15   20   25   30   35   40   50   55


menu driven program
1. To insert element at specific position
2. To insert element at beginning
3. To insert element at end
4. To insert an element before a specified index
5. To insert an element after the given index
6. To delete element at specific position
7. To delete element at beginning
8. To delete element at end
9. To delete an element before a specified index
10. To delete an element after the given index
11. To reverse the array
12. To end program

Enter you choice
7
The required result is--   15   20   25   30   35   40   50   55
```



```
Enter you choice
8
The required result is--   15   20   25   30   35   40   50


menu driven program
1. To insert element at specific position
2. To insert element at beginning
3. To insert element at end
4. To insert an element before a specified index
5. To insert an element after the given index
6. To delete element at specific position
7. To delete element at beginning
8. To delete element at end
9. To delete an element before a specified index
10. To delete an element after the given index
11. To reverse the array
12. To end program

Enter you choice
9
Enter the position
2
The required result is--   20   25   30   35   40   50
```



```
Enter you choice
10
Enter the position
4
The required result is--   20   25   30   35   50


menu driven program
1. To insert element at specific position
2. To insert element at beginning
3. To insert element at end
4. To insert an element before a specified index
5. To insert an element after the given index
6. To delete element at specific position
7. To delete element at beginning
8. To delete element at end
9. To delete an element before a specified index
10. To delete an element after the given index
11. To reverse the array
12. To end program

Enter you choice
11


reversed array--  50    35    30    25    20
```

****

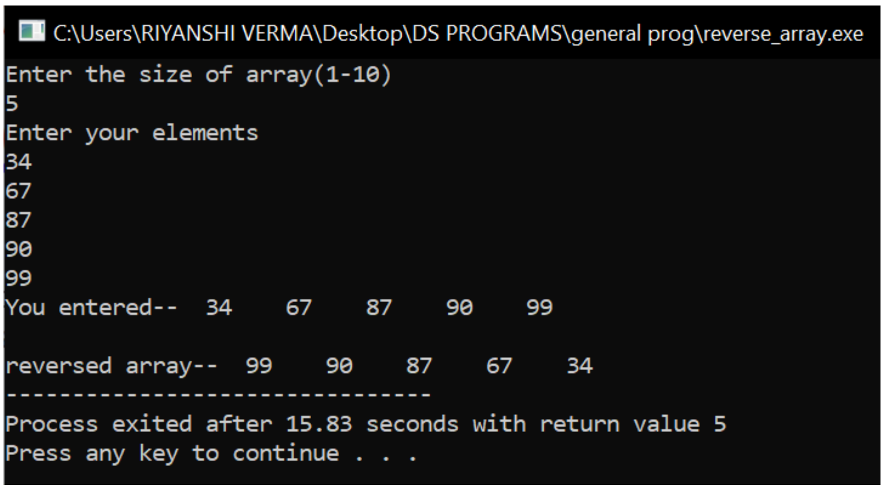
REVERSAL OF AN ARRAY

```
#include<stdlib.h>
void main()
{
	int n;
	printf("Enter the size of array(1-10)\n");
	scanf("%d",&n);
```

Riyanshi

Riyanshi

```
		if(n<0 || n>10)
		{
			printf("Enter correct value of n\n ");
			scanf("%d",&n);
		}
		if(n<0 || n>10)
		{
			exit(0);
		}
		int arr[100];
		int i;
		printf("Enter your elements\n");
		for(i=0;i<n;i++)
		{
			scanf("%d",&arr[i]);
		}
		printf("You entered--  ");
		for(i=0;i<n;i++)
		{
			printf("%d    ",arr[i]);
		}
		int temp;
		for(i=0;i<n/2;i++)
		{
			temp=arr[i];
			arr[i]=arr[n-1-i];
			arr[n-1-i]=temp;
		}
			printf("\n\nreversed array--  ");
		for(i=0;i<n;i++)
		{
			printf("%d    ",arr[i]);
		}
}
```


C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\reverse_array.exe

```
Enter the size of array(1-10)
5
Enter your elements
34
67
87
90
99
You entered--  34    67    87    90    99

reversed array--  99    90    87    67    34
--------------------------------
Process exited after 15.83 seconds with return value 5
Press any key to continue . . .
```

****

Riyanshi

Riyanshi

1913075

STACK USING SIMPLE ARRAY

```
#include <stdio.h>
#include <stdlib.h>
#define length 10
int top=-1;
int stack[length];
int isempty()
{
	 if(top==-1) return 1;
	 else return 0;
}
int isfull()
{
	if(top==length-1) return 1;
   else return 0;
}
void display()
{
	int i;
	if(isempty()==1)
	{
		printf("\n\nStack Underflow!!");
		return;
	}
	printf("\nStack--");
	for(i=top;i>=0;i--)
	printf("\n%d",stack[i]);
}
int peep()
{
	int temp;
	if(isempty()==1)
	{
		printf("\nStack Underflow");
		return(-78799);//return a value that does not belong to your data set
   }
	temp=stack[top];
	printf("Top remains %d",top);
	return (temp);
}
int pop()
{
	int temp;
	if(isempty()==1)
	{
		printf("\nStack Underflow");
		return(-78799);//return a value that does not belong to your data set
   }
	temp=stack[top];
```

Riyanshi

Riyanshi

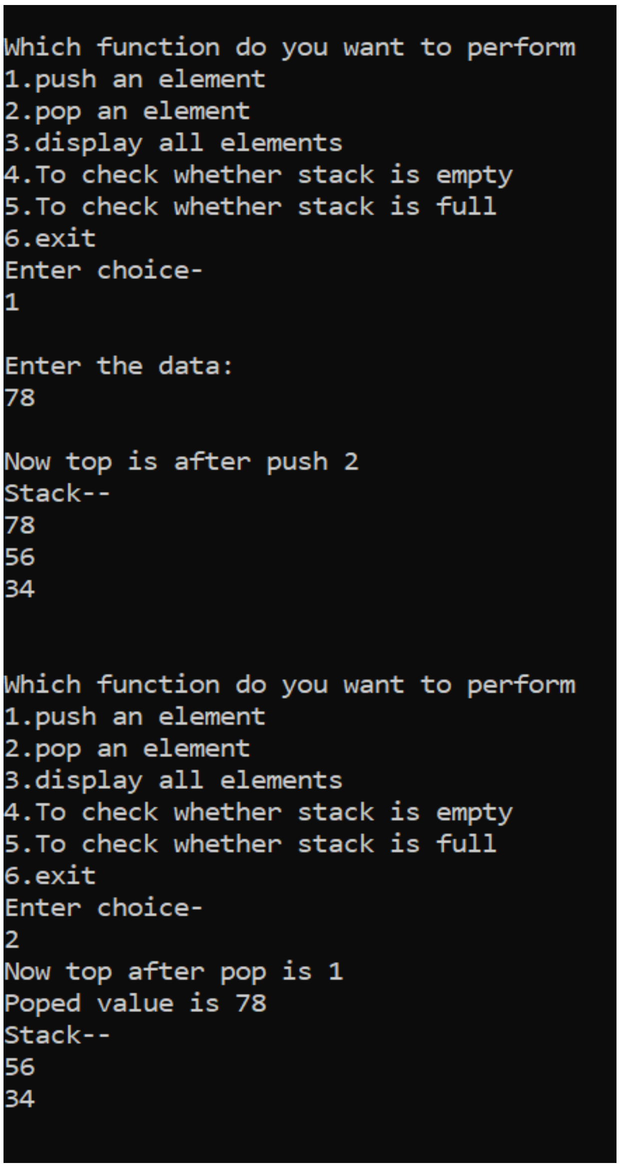
1913075

```
            top--;
            printf("Now top after pop is %d",top);
            return (temp);

}

void push()
{
            int item;
            if(isfull()==1)
            {
                        printf("\nStack Overflow!!");
            }
            else
            {
                        top++;
                        printf("\nEnter the data:\n");
                        scanf("%d",&item);
                        stack[top]=item;
                        printf("\nNow top is after push %d",top);
            }

}
int main()
{    int ch,i;int k;
            do{

            printf("\n\n\nWhich function do you want to perform");
            printf("\n1.push an element\n2.pop an element\n3.display all elements\n4.To check whether stack is
empty\n5.To check whether stack is full\n6.exit");
            printf("\nEnter choice-\n");

            scanf("%d",&ch);
            switch(ch)
            {
                        case 1: push();
                                 display();
                                  break;
                        case 2: i=pop();
                                  printf("\nPoped value is %d",i);
                                   display();
                                   break;
                        case 3: display();
                                    break;
                        case 4: k=isempty();
                                   if(k==1)
                                   printf("stack underflow/empty");
                                   else
                                   printf("stack has %d elements with top %d",top+1,top);
                                   break;
                         case 5: k=isfull();
                                  if(k==1)
```

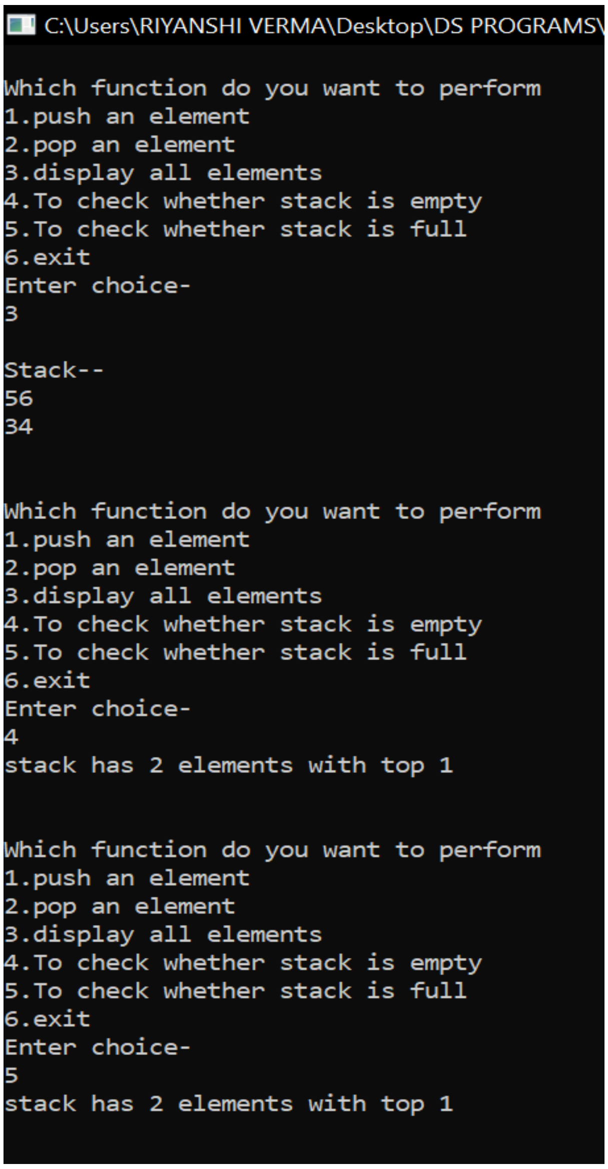
Riyanshi

Riyanshi

```
                        printf("stack overflow/full");
                        else
                        printf("stack has %d elements with top %d",top+1,top);
                        break;
                    case 6: exit(0);
                     default : printf("\nINPUT A VALID CHOICE");

        }
    }while(ch!=6);
}
```



```
Which function do you want to perform
1.push an element
2.pop an element
3.display all elements
4.To check whether stack is empty
5.To check whether stack is full
6.exit
Enter choice-
1

Enter the data:
78

Now top is after push 2
Stack--
78
56
34


Which function do you want to perform
1.push an element
2.pop an element
3.display all elements
4.To check whether stack is empty
5.To check whether stack is full
6.exit
Enter choice-
2
Now top after pop is 1
Poped value is 78
Stack--
56
34
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\

Which function do you want to perform
1.push an element
2.pop an element
3.display all elements
4.To check whether stack is empty
5.To check whether stack is full
6.exit
Enter choice-
3

Stack--
56
34


Which function do you want to perform
1.push an element
2.pop an element
3.display all elements
4.To check whether stack is empty
5.To check whether stack is full
6.exit
Enter choice-
4
stack has 2 elements with top 1


Which function do you want to perform
1.push an element
2.pop an element
3.display all elements
4.To check whether stack is empty
5.To check whether stack is full
6.exit
Enter choice-
5
stack has 2 elements with top 1
```

****

Riyanshi

Riyanshi

1913075

STACK USING STRUCTURES

```
#include <stdio.h>
#include <stdlib.h>
#define length 10
struct st
{
int top;
int stack[length];
}s1;

void display()
{
        int i;
        if(s1.top==-1)
        {
                printf("\n\nStack Underflow!!");
                return;
        }
        printf("\nStack--");
        for(i=s1.top;i>=0;i--)
        printf("\n%d",s1.stack[i]);
}
int pop()
{
        int temp;
        if(s1.top==-1)
        {
                printf("\nStack Underflow");
                return(-78799);//return a value that does not belong to your data set
  }
        temp=s1.stack[s1.top];
        s1.top--;
        printf("Now top after pop is %d",s1.top);
        return (temp);

}

void push()
{
        int item;
        if(s1.top==length-1)
        {
                printf("\nStack Overflow!!");
        }
        else
        {
                s1.top++;
                printf("\nEnter the data:\n");
                scanf("%d",&item);
                s1.stack[s1.top]=item;
                printf("\nNow top is after push %d",s1.top);
```

Riyanshi

Riyanshi

1913075

```
            }

}
int main()
{    int ch,i;
s1.top=-1;
            do{

            printf("\n\n\nWhich function do you want to perform");
            printf("\n1.push an element\n2.pop an element\n3.display all elements\n4.exit");
            printf("\nEnter choice-\n");

            scanf("%d",&ch);
            switch(ch)
            {
                        case 1: push();
                                  display();
                                   break;
                        case 2: i=pop();
                                   printf("\nPoped value is %d",i);
                                   display();
                                   break;
                        case 3: display();
                                    break;
                        case 4: exit(0);
                        default : printf("\nINPUT A VALID CHOICE");

            }
     }while(ch!=4);
}
```

OUTPUT SAME AS PREVIOUS(STACK USING SIMPLE ARRAY)

****

STACK USING POINTERS WITH STRUCTURES

```
#include <stdio.h>
#include <stdlib.h>
#define length 10
struct stack
{
int top;
int stack[length];
}stack;

void display(struct stack *s)
{
            int i;
            if(s->top==-1)
            {
```

Riyanshi

Riyanshi

```
                printf("\n\nStack Underflow!!");
                return;
        }
        printf("\nStack--");
        for(i=s->top;i>=0;i--)
        printf("\n%d",s->stack[i]);
}
int pop(struct stack *s)
{
        int temp;
        if(s->top==-1)
        {
                printf("\nStack Underflow");
                return(-78799);//return a value that does not belong to your data set
   }
        temp=s->stack[s->top];
        s->top--;
        printf("Now top after pop is %d",s->top);
        return (temp);

}

void push(struct stack *s)
{
        int item;
        if(s->top==length-1)
        {
                printf("\nStack Overflow!!");
        }
        else
        {
                s->top++;
                printf("\nEnter the data:\n");
                scanf("%d",&item);
                s->stack[s->top]=item;
                printf("\nNow top is after push %d",s->top);
        }

}
int main()
{   int ch,i;
struct stack stk;
stk.top=-1;
        do{

        printf("\n\n\nWhich function do you want to perform");
        printf("\n1.push an element\n2.pop an element\n3.display all elements\n4.exit");
        printf("\nEnter choice-\n");

        scanf("%d",&ch);
        switch(ch)
        {
```

Riyanshi

Riyanshi

```
                case 1: push(&stk);
                            display(&stk);
                             break;
                case 2: i=pop(&stk);
                            printf("\nPoped value is %d",i);
                            display(&stk);
                            break;
                case 3: display(&stk);
                            break;
                case 4: exit(0);
                default : printf("\nINPUT A VALID CHOICE");

            }
    }while(ch!=4);
}
```

OUTPUT SAME AS PREVIOUS(STACK USING SIMPLE ARRAY)

****

MULTISTACK

```
#include<stdio.h>
#include<string.h>
#include<process.h>
#define MAX 10
struct stack
{
            int data[MAX];
            int topA,topB;
}s;
void displayA()
{
            int i;
            if(s.topA==-1)
            {
                        printf("\n\nStackA Underflow!!");
                        return;
            }
            printf("\nStack--");
            for(i=s.topA;i>=0;i--)
            printf("\n%d",s.data[i]);
            if(s.topA==4)
            printf("\nStackA overflow!!");
}
void displayB()
{
            int i;
            if(s.topB==4)
```

Riyanshi

Riyanshi

```
            {
                        printf("\n\nStackB Underflow!!");
                        return;
            }
            printf("\nStack--");
            for(i=s.topB;i>=5;i--)
            {
                        printf("\n%d",s.data[i]);
            }
            if(s.topB==MAX-1)
            printf("\nStackB overflow!!");

}
void pushA(int x)
{

            if (s.topA>=-1 && s.topA<4)
            {
        s.topA++;
        s.data[s.topA] = x;
    }
    else
        printf("StackA Overflow");
}
void pushB(int x)
{
            if (s.topB >= 4 && s.topB<MAX)
            {
        s.topB++;
        s.data[s.topB] =x;
    }
    else
        printf("StackB Overflow!!");
}
int popA()
    {
        if (s.topA >= 0)
                        {
            int x = s.data[s.topA];
                                    s.topA--;
            return x;
        }
        else {
             printf("StackA UnderFlow") ;
             return -799979;
                        }

    }
    int popB()
    {
        if (s.topB >= 5) {
            int x = s.data[s.topB];
```

Riyanshi

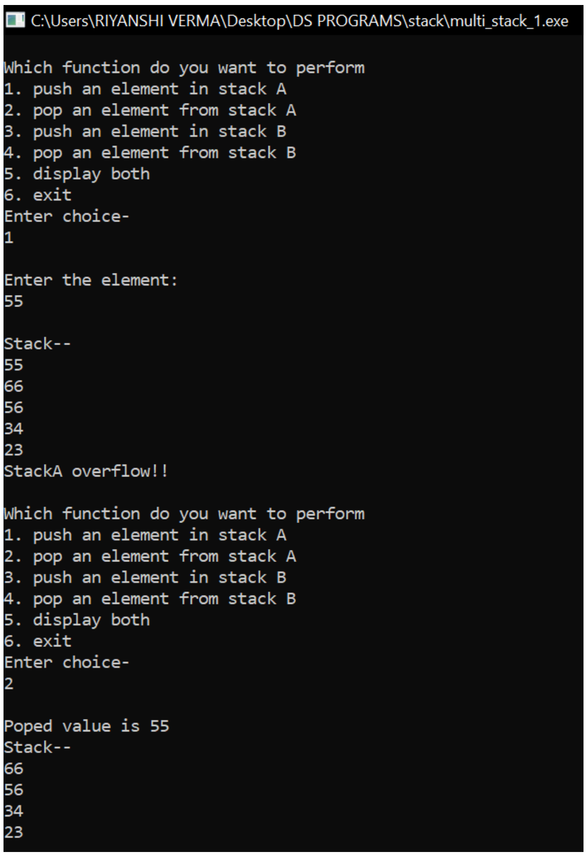
Riyanshi

```
        s.topB--;
        return x;
      }
      else
                  {
          printf("StackB UnderFlow") ;
                          return -799979 ;
                  }

   }

int main()
{   int ch,i,num;int k;s.topA=-1;s.topB=4;
          do{

          printf("\n\nWhich function do you want to perform");
          printf("\n1. push an element in stack A\n2. pop an element from stack A\n3. push an element in stack 
B\n4. pop an element from stack B\n5. display both\n6. exit");
          printf("\nEnter choice-\n");

          scanf("%d",&ch);
          switch(ch)
          {
                  case 1:  printf("\nEnter the element:\n");
                           scanf("%d",&num);
                           pushA(num);
                           displayA();
                           break;
                  case 2: i=popA();
                          printf("\nPoped value is %d",i);
                          displayA();
                          break;
                  case 3: printf("\nEnter the element:\n");
                          scanf("%d",&num);
                          pushB(num);
                         displayB();
                         break;
                  case 4: i=popB();
                         printf("\nPoped value is %d",i);
                         displayB();
                         break;
                  case 5: printf("\nStack A:  ");
                          displayA();
                          printf("\nStack B:  ");
                           displayB();
                           break;
                   case 6: exit(0);
                   default : printf("\nINPUT A VALID CHOICE");

          }
     }while(ch!=6);
}
```

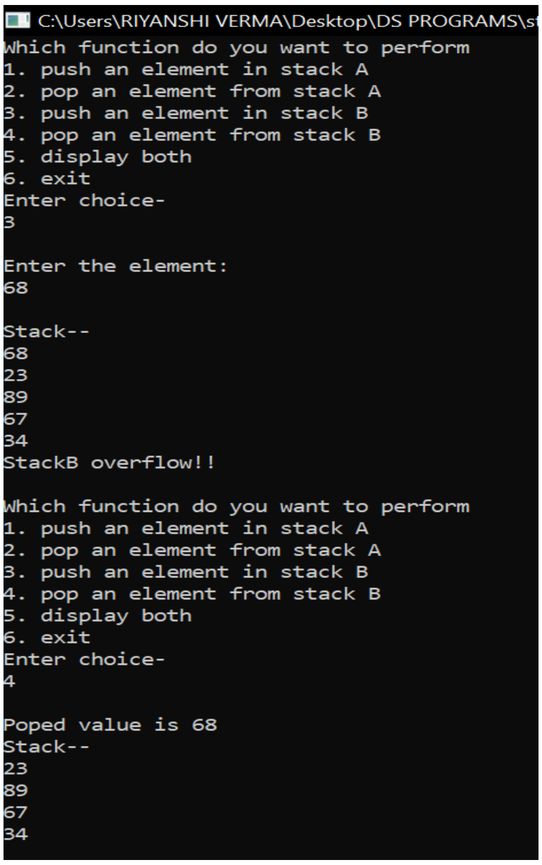
Riyanshi

Riyanshi

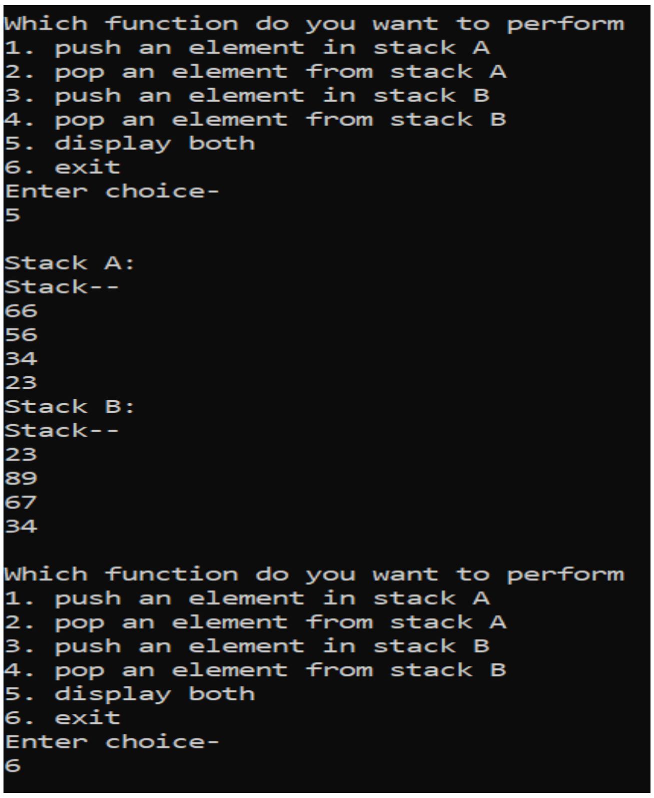
1913075



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\stack\multi_stack_1.exe

Which function do you want to perform
1. push an element in stack A
2. pop an element from stack A
3. push an element in stack B
4. pop an element from stack B
5. display both
6. exit
Enter choice-
1

Enter the element:
55

Stack--
55
66
56
34
23
StackA overflow!!

Which function do you want to perform
1. push an element in stack A
2. pop an element from stack A
3. push an element in stack B
4. pop an element from stack B
5. display both
6. exit
Enter choice-
2

Poped value is 55
Stack--
66
56
34
23
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\s
Which function do you want to perform
1. push an element in stack A
2. pop an element from stack A
3. push an element in stack B
4. pop an element from stack B
5. display both
6. exit
Enter choice-
3

Enter the element:
68

Stack--
68
23
89
67
34
StackB overflow!!

Which function do you want to perform
1. push an element in stack A
2. pop an element from stack A
3. push an element in stack B
4. pop an element from stack B
5. display both
6. exit
Enter choice-
4

Poped value is 68
Stack--
23
89
67
34
```



```
Which function do you want to perform
1. push an element in stack A
2. pop an element from stack A
3. push an element in stack B
4. pop an element from stack B
5. display both
6. exit
Enter choice-
5

Stack A:
Stack--
66
56
34
23
Stack B:
Stack--
23
89
67
34

Which function do you want to perform
1. push an element in stack A
2. pop an element from stack A
3. push an element in stack B
4. pop an element from stack B
5. display both
6. exit
Enter choice-
6
```

****

Riyanshi

Riyanshi

INFIX TO PREFIX

```
#include<stdio.h>
#include<string.h>
#include<process.h>
#define MAX 100
char stack[MAX];
int top=-1;
char pop()
{
        char a;
        a=stack[top];
        top--;
        return a;
}
void push(char item)
{
        top++;
        stack[top]=item;
}
int prcd(char sym)
{
        switch(sym)
        {
                case '+':
                case '-': return 2;
                case '*':
                case '%':
                case '/': return 4;
                case '^':
                case '$': return 6;
                case '(':
                case ')': return 1;
        }
}
int isoperand(char sym)
{
        if((sym>='a' && sym<='z') || (sym>='A' && sym<='Z') || (sym>='0' && sym<='9'))
        return 1;
        else
        return 0;
}
int isoperator(char sym)
{
        switch(sym)
        {
                case '+':
                case '-':
                case '*':
                case '%':
                case '/':
                case '^':
```

Riyanshi

Riyanshi

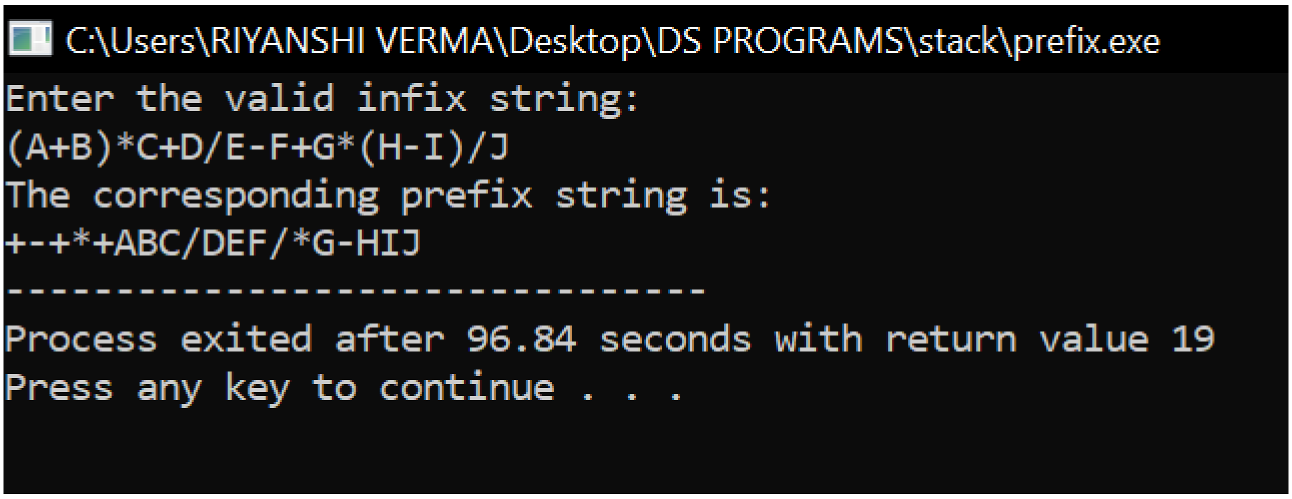
1913075

```
            case '$':
            case '(':
            case ')': return 1;
            default : return 0;
        }
}
void convert_to_pre(char infix[],char prefix[])
{
        int i,sym,j=0;
        strrev(infix);
        for(i=0;i<strlen(infix);i++)
        {
            sym=infix[i];
            if(isoperand(sym)==1)
            {
                    prefix[j]=sym;j++;
            }
            else if(sym==')')
            push(sym);
            else if(sym=='(')
            {
                    while(stack[top]!=')')
                    {
                            prefix[j]=pop();
                            j++;
                    }
                    pop();
            }
            else if(isoperator(sym)==1)
            {
                    if(prcd(sym)>=prcd(stack[top]))
                    push(sym);
                    else
                    {
                            while(prcd(sym)<prcd(stack[top]))
                            {
                                    prefix[j]=pop();
                                    j++;
                            }
                            push(sym);
                    }
            }
            else
            {
                    printf("\nINVALID : %c",sym);
                    exit(0);
            }
        }
        while(top!=-1)
        {
            prefix[j]=pop();
            j++;
```

Riyanshi

Riyanshi

```
            }
            prefix[j]='\0';
}
void main()
{
            char infix[100],prefix[100];
            printf("Enter the valid infix string:\n");
            scanf("%s",infix);
            convert_to_pre(infix,prefix);
   strrev(prefix);
            printf("The corresponding prefix string is:\n");
            printf("%s",prefix);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\stack\prefix.exe
Enter the valid infix string:
(A+B)*C+D/E-F+G*(H-I)/J
The corresponding prefix string is:
+-+*+ABC/DEF/*G-HIJ
--------------------------------
Process exited after 96.84 seconds with return value 19
Press any key to continue . . .
```

****

INFIX TO POSTFIX

```
#include<stdio.h>
#include<string.h>
#include<process.h>
#define MAX 100
char stack[MAX];
int top=-1;
char pop()
{
            char a;
            a=stack[top];
            top--;
            return a;
}
void push(char item)
{
            top++;
            stack[top]=item;
}
```

Riyanshi

Riyanshi

1913075

```
int prcd(char sym)
{
	switch(sym)
	{
		case '+':
		case '-': return 2;
		case '*':
		case '%':
		case '/': return 4;
		case '^':
		case '$': return 6;
		case '(':
		case ')': return 1;
	}
}
int isoperand(char sym)
{
	if((sym>='a' && sym<='z') || (sym>='A' && sym<='Z') || (sym>='0' && sym<='9'))
	return 1;
	else
	return 0;
}
int isoperator(char sym)
{
	switch(sym)
	{
		case '+':
		case '-':
		case '*':
		case '%':
		case '/':
		case '^':
		case '$':
		case '(':
		case ')': return 1;
		default : return 0;
	}
}
void convert_to_post(char infix[],char postfix[])
{
	int i,sym,j=0;
	for(i=0;i<strlen(infix);i++)
	{
		sym=infix[i];
		if(isoperand(sym)==1)
		{
			postfix[j]=sym;j++;
		}
		else if(sym=='(')
		push(sym);
		else if(sym==')')
		{
```

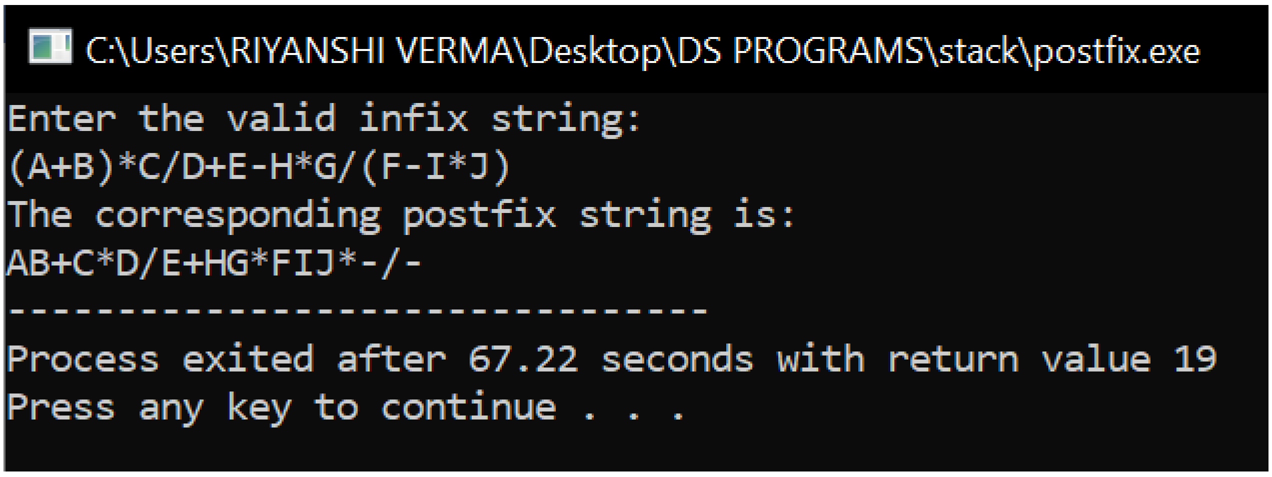
Riyanshi

Riyanshi

```
                        while(stack[top]!='(')
                        {
                                postfix[j]=pop();
                                j++;
                        }
                        pop();
                }
                else if(isoperator(sym)==1)
                {
                        if(prcd(sym)>prcd(stack[top]))
                        push(sym);
                        else
                        {
                                while(prcd(sym)<=prcd(stack[top]))
                                {
                                        postfix[j]=pop();
                                        j++;
                                }
                                push(sym);
                        }
                }
                else
                {
                        printf("\nINVALID : %c",sym);
                        exit(0);
                }
        }
        while(top!=-1)
        {
                postfix[j]=pop();
                j++;
        }
        postfix[j]='\0';
}
void main()
{
        char infix[100],postfix[100];
        printf("Enter the valid infix string:\n");
        scanf("%s",infix);
        convert_to_post(infix,postfix);
        printf("The corresponding postfix string is:\n");
        printf("%s",postfix);
}
```

Riyanshi

Riyanshi



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\stack\postfix.exe
Enter the valid infix string:
(A+B)*C/D+E-H*G/(F-I*J)
The corresponding postfix string is:
AB+C*D/E+HG*FIJ*-/-
--------------------------------
Process exited after 67.22 seconds with return value 19
Press any key to continue . . .
```

****

POSTFIX EVALUATION

```
#include<stdio.h>
#include<string.h>
#include<process.h>
#define MAX 20
struct stack
{
        int data[MAX];
        int top;
}s;
int pop()
{
        int ret;
        if(s.top==-1)
        {
                printf("\nSTACK EMPTY");
                return -111;
        }
        else
        {
                ret=s.data[s.top];
                --s.top;
                return ret;
        }
}

void push(int item)
{
        if(s.top==(MAX-1))
        printf("\nSTACK FULL");
        else
        {
                ++s.top;
                s.data[s.top]=item;
        }
}
```

Riyanshi

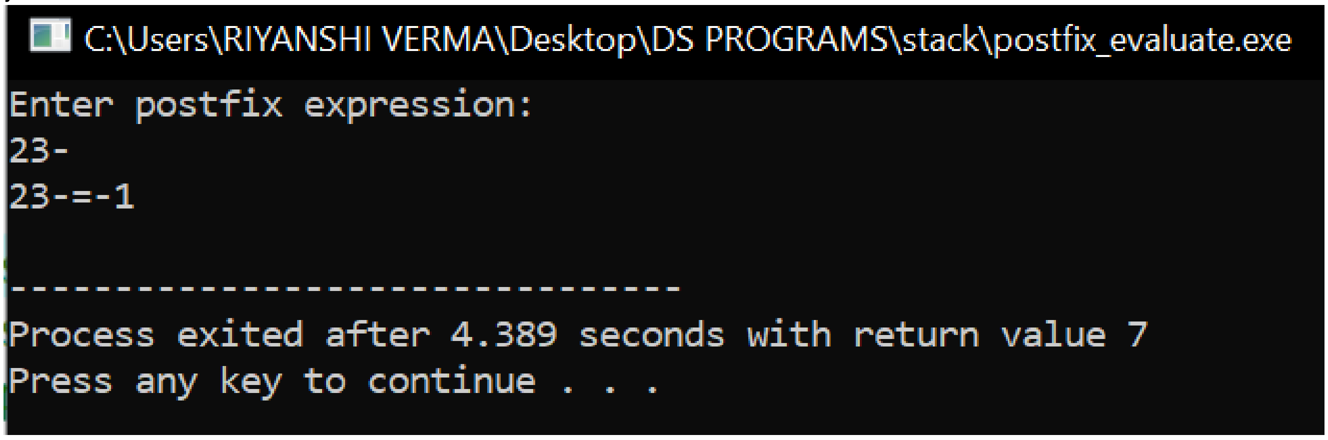
Riyanshi

```
int isdigits(char c)
{
        if(c>='0' && c<='9')
        return 1;
        else
        return 0;
}
int isoperator(char sym)
{
        switch(sym)
        {
                case '+':
                case '-':
                case '*':
                case '%':
                case '/':
                case '^':
                case '$':
                case '(':
                case ')': return 1;
                default : return 0;
        }
}
int evaluate(char postfix[])
{
        int i; char sym;
        int op1,op2,result;
        for(i=0;i<strlen(postfix);i++)
        {
                sym=postfix[i];
                while(sym==' ' || sym=='\t')
                {
                        i++;
                }

        if(isdigits(sym))
        {
                push(sym-48);
        }
        else if(isoperator(sym))
        {
                op1=pop();
                op2=pop();
                switch(sym)
                {
                        case '+' : result=op1+op2;break;
                        case '-' : result=op2-op1;break;
                        case '/' : result=op2/op1;break;
                        case '*' : result=op1*op2;break;
                        case '%' : result=op2%op1;break;
                }
                push(result);
```

Riyanshi

Riyanshi

```
		}
		else
		{
			printf("\nINVALID");exit(0);
		}
	}
		result=pop();
		return result;
}
void main()
{
		char exp[MAX];
		int ans;
		s.top=-1;
		printf("Enter postfix expression:\n");
		scanf("%s",exp);
		ans=evaluate(exp);
		printf("%s=%d\n",exp,ans);
}
```



C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\stack\postfix_evaluate.exe

```
Enter postfix expression:
23-
23-=-1

--------------------------------
Process exited after 4.389 seconds with return value 7
Press any key to continue . . .
```

****

REVERSAL OF STRING USING STACK

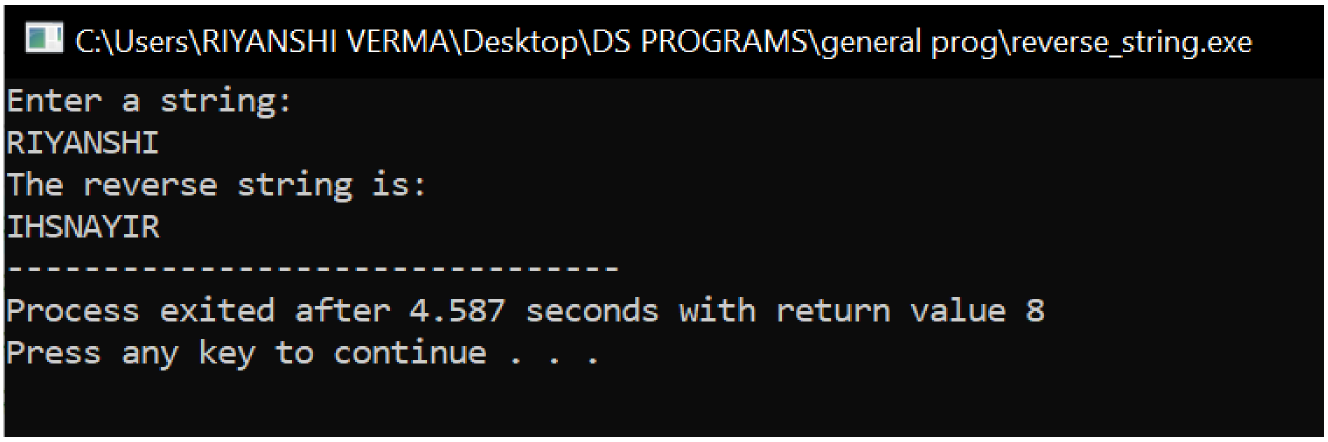
```
#include<stdio.h>
#include<string.h>
#include<process.h>
#define MAX 20
char stack[MAX];
int top=-1;
char pop()
{
		char a;
		a=stack[top];
		top--;
		return a;
}
void push(int item)
{
```

Riyanshi

Riyanshi

1913075

```
	if(top==MAX-1)
	{
		printf("Stack overflow");
		return;
	}
	top++;
	stack[top]=item;
}
void reverse(char* str)
{
	int i,len;
	len=strlen(str);
	for(i=0;i<len;i++)
	push(str[i]);
	for(i=0;i<len;i++)
	str[i]=pop();
}
void main()
{
	char str[20];
	printf("Enter a string:\n");
	scanf("%s",str);
	reverse(str);
	printf("The reverse string is:\n");
	printf("%s",str);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\reverse_string.exe
Enter a string:
RIYANSHI
The reverse string is:
IHSNAYIR
--------------------------------
Process exited after 4.587 seconds with return value 8
Press any key to continue . . .
```

****

PARANTHESIS CHECKER

```
#include<stdio.h>
#include<string.h>
#define MAX 20

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

void push(char symbol){
	top++;
```

Riyanshi

Riyanshi

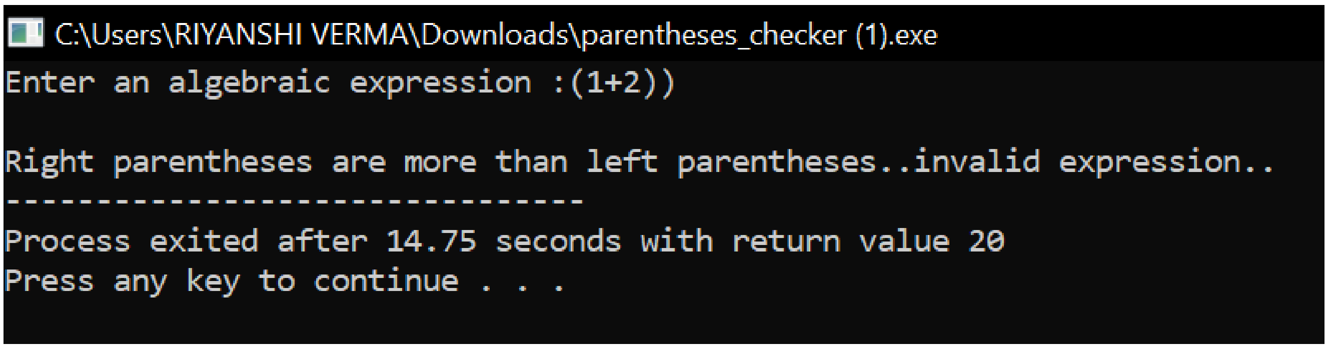
```
        stack[top]=symbol;
        return;
}

char pop(){
        char a = stack[top];
        top--;
        return a;
}
int match(char a,char b){
        if(a=='[' &&b==']')
                return 1;
        if(a=='{' && b=='}')
                return 1;
        if(a=='(' && b==')')
                return 1;
        return 0;
}

int isBalanced(char exp[]){
        int i;
        char temp;
        for(i=0;i<strlen(exp);i++){
                if(exp[i]=='('||exp[i]=='['||exp[i]=='{')
                        push(exp[i]);
                else if(exp[i]==')'||exp[i]==']'||exp[i]=='}'){
                        if(top==-1){
                                printf("\nRight parentheses are more than left parentheses..");
                                return 0;
                        }
                        else{
                                temp=pop();
                                if(!match(temp,exp[i])){
                                        printf("\nMismatched parentheses are : ");
                                        printf("%c and %c ",temp,exp[i]);
                                        return 0;
                                }
                        }
                }
        }
        if(top!=-1){
                printf("\nLeft parentheses more than right parentheses.. ");
                return 0;
        }
        else{
                printf("\nBalanced parentheses..");
                return 1;
        }
}
void main(){
        char exp[MAX];
        int valid;
```

Riyanshi

Riyanshi

1913075

```
        printf("Enter an algebraic expression :");
        gets(exp);
        valid = isBalanced(exp);
        if(valid==1){
                printf("valid expression..");
        }
        else{
                printf("invalid expression..");
        }
}
```



```
C:\Users\RIYANSHI VERMA\Downloads\parentheses_checker (1).exe
Enter an algebraic expression :(1+2))

Right parentheses are more than left parentheses..invalid expression..
--------------------------------
Process exited after 14.75 seconds with return value 20
Press any key to continue . . .
```

****

SIMPLE QUEUE

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10
int queue[MAX];
int rear = - 1;
int front = - 1;
int size()
{
        int i,c=0;
        for(i=front;i<=rear;i++)
        c++;
        if(rear==-1)
        return 0;
        else
        return c;
}
void enqueue()
{
  int item;
  if (rear == MAX - 1)
  printf("Queue Overflow \n");
  else
  {
    if (front == - 1)

    front = 0;
    printf("\nEnter the element to be inserted in queue : \n");
```

Riyanshi

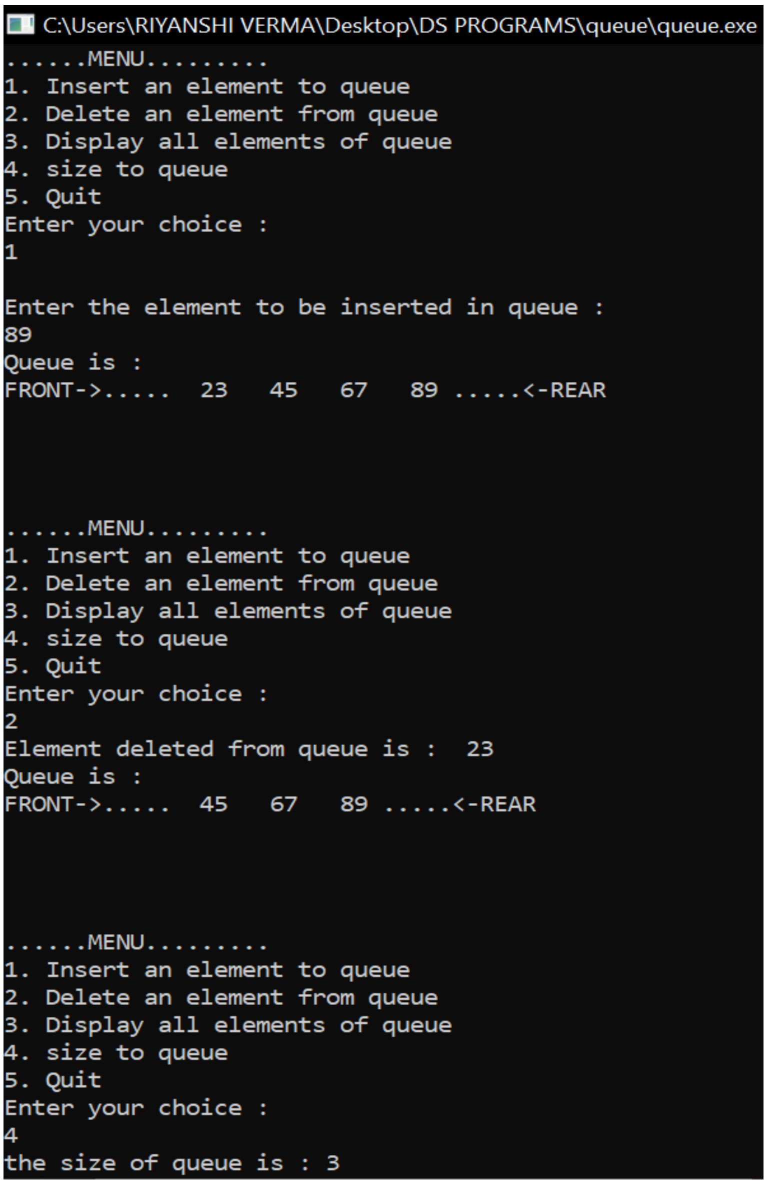
Riyanshi

```
        scanf("%d", &item);
        rear = rear + 1;
        queue[rear] =item;
    }
}

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

void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty(underflow)\n");
    else
    {
        printf("Queue is : \nFRONT->.....");
        for (i = front; i <= rear; i++)
            printf("  %d ", queue[i]);
        printf(".....<-REAR\n");
    }
}
void main()
{
    int choice,i;
    do
    {
        printf("\n\n\n\n......MENU.........\n");
        printf("1. Insert an element to queue \n");
        printf("2. Delete an element from queue \n");
        printf("3. Display all elements of queue \n");
        printf("4. size to queue\n");
        printf("5. Quit \n");
        printf("Enter your choice : \n");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1: enqueue();display();
                    break;
            case 2: dequeue();display();
                    break;
```

Riyanshi

Riyanshi

```
        case 3: display();
              break;
        case 4: i=size();
              printf("the size of queue is : %d",i);
              break;
        case 5: printf("THANKYOU");
                                exit(0);
                                              break;
        default: printf("Wrong choice \n");
      }
   }while(choice!=5);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\queue\queue.exe
......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
1

Enter the element to be inserted in queue :
89
Queue is :
FRONT->.....  23   45   67   89 .....<-REAR




......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
2
Element deleted from queue is :  23
Queue is :
FRONT->.....  45   67   89 .....<-REAR




......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
4
the size of queue is : 3
```

****

PRIORITY QUEUE USING 2-D ARRAY

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

Riyanshi

Riyanshi

1913075

```
#define MAX 5
int queue[MAX][MAX];
int rear[MAX];
int front [MAX];
void enqueue()
{
   int item,p,r;
   printf("\nEnter the priority of element\n");
   scanf("%d",&p);
   if (rear[p] == MAX-1)
            printf("Queue of %d priority Overflow \n",p);
   else
   {
            printf("\nEnter elements to enqueue\n");
           scanf("%d", &item);
                    if (front[p] == - 1)
                    {
                             front[p]=0;
                    }
           rear[p]++;
          r=rear[p];
       queue[p][r]=item;
      printf("\nRear = %d  Front = %d",rear[p],front[p]);
   }
}

void dequeue()
{
          int temp=-111,flag=0,i,f;
          for(i=MAX-1;i>=0;i--)//index more priority moree
   {
             if (front[i]!=-1)
      {
          f=front[i];
          temp=queue[i][f];
          if(front[i]==rear[i])
          front[i]=rear[i]=-1;
          else
          front[i]++;
          flag=1;break;
      }
   }
   if(flag==0)
      printf("All Queue are Underflow \n");
   else
      printf("\nElements dequeue=%d with priority =%d",temp,i);
}

void display()
{
   int i,j;
   printf("\nRear array=  ");
```

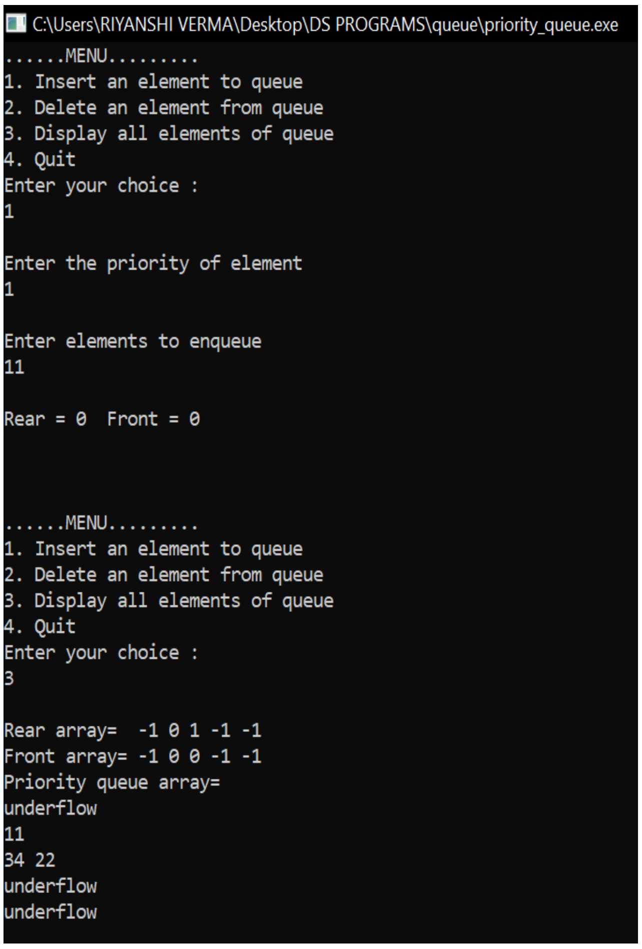
Riyanshi

Riyanshi

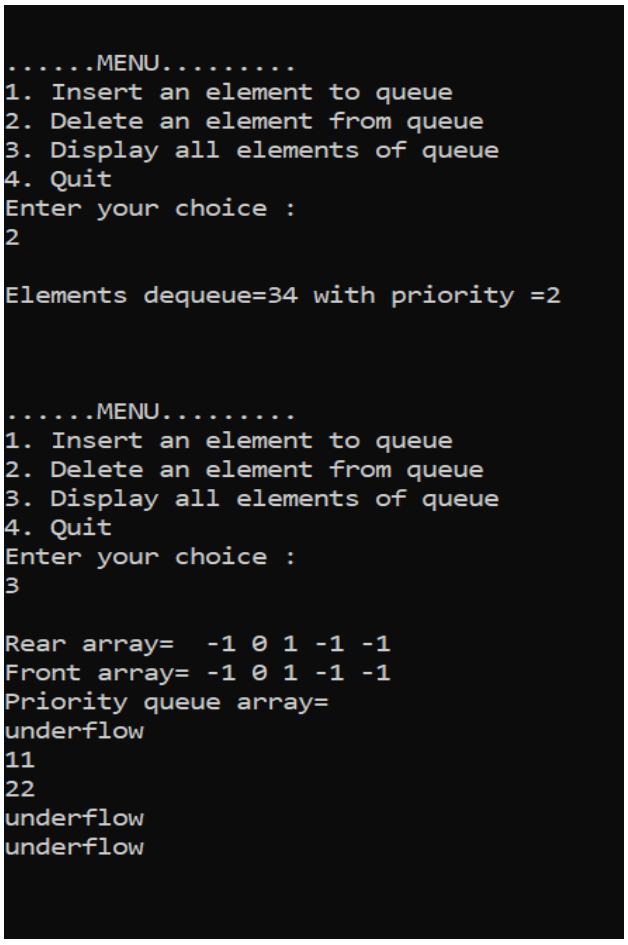
```
   for(i=0;i<MAX;i++)
   printf("%d ",rear[i]);
   printf("\nFront array= ");
   for(i=0;i<MAX;i++)
   printf("%d ",front[i]);
   printf("\nPriority queue array=\n");
   for(i=0;i<MAX;i++)
   {
           for(j=front[i];j<=rear[i];j++)
           {
                       if(j==-1)
                       printf("underflow");
                       else
                       printf("%d ",queue[i][j]);

                       }
                       printf("\n");

           }
}
void main()
{

   int choice,i;
   for(i=0;i<MAX;i++)
   rear[i]=front[i]=-1;
   do
   {
      printf("\n\n\n\n......MENU.........\n");
      printf("1. Insert an element to queue \n");
      printf("2. Delete an element from queue \n");
      printf("3. Display all elements of queue \n");
      printf("4. Quit \n");
      printf("Enter your choice : \n");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1: enqueue();
               break;
         case 2: dequeue();
               break;
         case 3: display();
               break;
         case 4: printf("THANKYOU");
                                   exit(0);
                                                  break;
         default: printf("Wrong choice \n");
      }
   }while(choice!=5);
}
```

Riyanshi

Riyanshi



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\queue\priority_queue.exe
......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
1

Enter the priority of element
1

Enter elements to enqueue
11

Rear = 0  Front = 0



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
3

Rear array=  -1 0 1 -1 -1
Front array= -1 0 0 -1 -1
Priority queue array=
underflow
11
34 22
underflow
underflow
```



```
......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
2

Elements dequeue=34 with priority =2



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
3

Rear array=  -1 0 1 -1 -1
Front array= -1 0 1 -1 -1
Priority queue array=
underflow
11
22
underflow
underflow
```

****

PRIORITY QUEUE HAVING EXPENSIVE INSERTION

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int queue[MAX];
int rear = - 1;
int front = - 1;
int size()
{
	int i,c=0;
	for(i=front;i<=rear;i++)
	c++;
	if(rear==-1)
	return 0;
	else
	return c;
}
void sort()
{
	int j,x,i;
	for(i=front+1;i<=rear;i++)
	{
		j=i;x=queue[i];
```

Riyanshi

Riyanshi

```
                while(queue[j-1]>x && j>0)
                {
                        queue[j]=queue[j-1];
                        j=j-1;
                }
                if (j!=i)
                queue[j]=x;
        }
        printf("\n\nThe sorted list of elements is--  ");
        for(i=front;i<=rear;i++)
        {
                printf("%d    ",queue[i]);
        }

}
void shift()
{
        int i;
        for(i=front;i<=rear;i++)
        {
                queue[i-1]=queue[i];
        }
        front=0;rear=rear-1;
}
void enqueue()
{
  int item;
  if (rear == MAX - 1)
  printf("Queue Overflow \n");
  else
  {
    if (front == - 1)

    front = 0;
    printf("\nEnter the element to be inserted in queue : \n");
    scanf("%d", &item);
    rear = rear + 1;
    queue[rear] =item;
    sort();


  }
}

void dequeue()
{
  if (front == - 1 || front > rear)
  {
    printf("Queue Underflow \n");
    return ;
  }
  else
```

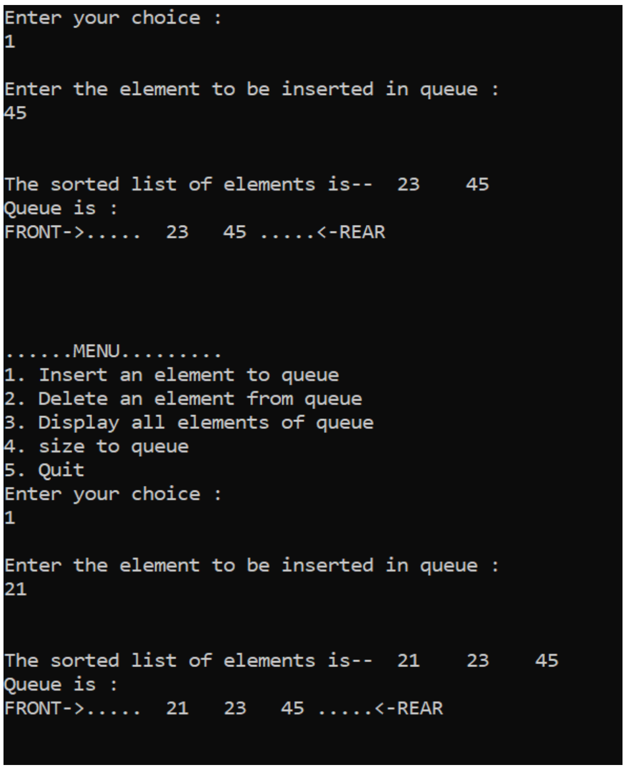
Riyanshi

Riyanshi

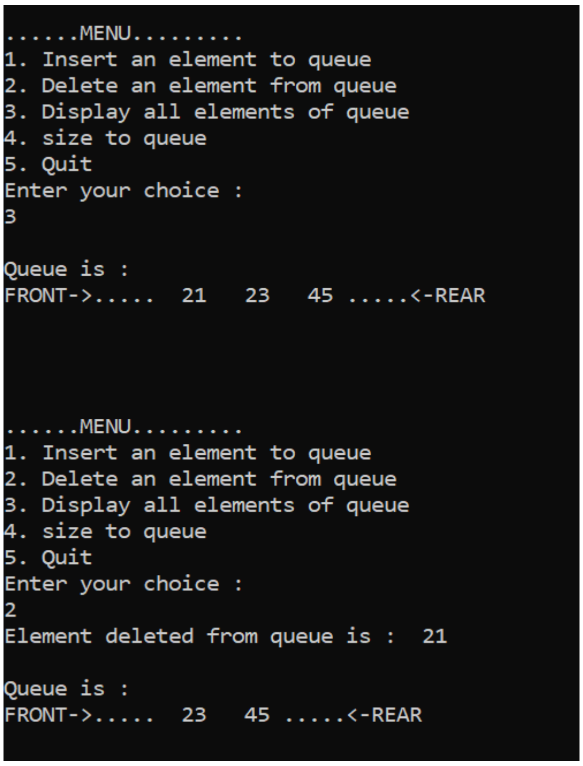
```
   {
      printf("Element deleted from queue is :  %d\n", queue[front]);
      front = front + 1;
   }
}

void display()
{
   int i;
   if (front == - 1)
      printf("Queue is empty(underflow)\n");
   else
   {
      printf("\nQueue is : \nFRONT->.....");
      for (i = front; i <= rear; i++)
         printf("  %d ", queue[i]);
      printf(".....<-REAR\n");
   }
}
void main()
{
   int choice,i;
   do
   {
      printf("\n\n\n\n......MENU.........\n");
      printf("1. Insert an element to queue \n");
      printf("2. Delete an element from queue \n");
      printf("3. Display all elements of queue \n");
      printf("4. size to queue\n");
      printf("5. Quit \n");
      printf("Enter your choice : \n");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1: enqueue();display();
               break;
         case 2: dequeue();shift();display();
               break;
         case 3: display();
               break;
         case 4: i=size();
               printf("the size of queue is : %d",i);
               break;
         case 5: printf("THANKYOU");
                                   exit(0);
                                                break;
         default: printf("Wrong choice \n");
      }
   }while(choice!=5);
}
```

Riyanshi

Riyanshi



```
Enter your choice :
1

Enter the element to be inserted in queue :
45


The sorted list of elements is--   23    45
Queue is :
FRONT->.....  23   45 .....<-REAR




......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
1

Enter the element to be inserted in queue :
21


The sorted list of elements is--   21    23    45
Queue is :
FRONT->.....  21   23   45 .....<-REAR
```



```
......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
3

Queue is :
FRONT->.....  21   23   45 .....<-REAR




......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
2
Element deleted from queue is :  21

Queue is :
FRONT->.....  23   45 .....<-REAR
```

****

PRIORITY QUEUE HAVING EXPENSIVE DELETION

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int queue[MAX];
int rear = - 1;
int front = - 1;
int size()
{
        int i,c=0;
        for(i=front;i<=rear;i++)
        c++;
        if(rear==-1)
        return 0;
        else
        return c;
}
void shift()
{
        int i;
        for(i=front;i<=rear;i++)
        {
                queue[i-1]=queue[i];
        }
        front=0;rear=rear-1;
}
void sort()
```

Riyanshi

Riyanshi

```
{
		int i,j,temp;
		for(i=0;i<=rear-1;i++)
		{
			for(j=0;j<=(rear-i-1);j++)
			{
				if(queue[j]>queue[j+1])
				{
					temp=queue[j];
					queue[j]=queue[j+1];
					queue[j+1]=temp;

				}
			}
		}
		printf("\n\nThe elements are--  ");
		for(i=0;i<=rear;i++)
		{
			printf("%d    ",queue[i]);
		}
}
void enqueue()
{
   int item;
   if (rear == MAX - 1)
   printf("Queue Overflow \n");
   else
   {
      if (front == - 1)

      front = 0;
      printf("\nEnter the element to be inserted in queue : \n");
      scanf("%d", &item);
      rear = rear + 1;
      queue[rear] = item;

   }
}

void dequeue()
{
		sort();
   if (front == - 1 || front > rear)
   {
      printf("Queue Underflow \n");
      return ;
   }
   else
   {
      printf("\nElement deleted from queue is :  %d\n", queue[front]);
      front = front + 1;
```

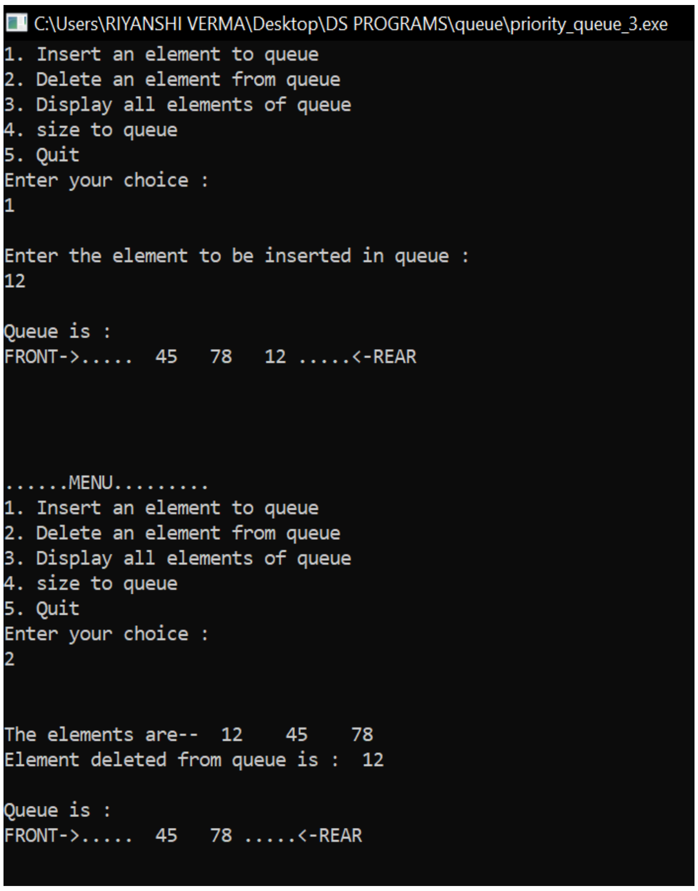
Riyanshi

Riyanshi

```
   }
}

void display()
{
   int i;
   if (front == - 1)
      printf("Queue is empty(underflow)\n");
   else
   {
      printf("\nQueue is : \nFRONT->.....");
      for (i = front; i <= rear; i++)
         printf("  %d ", queue[i]);
      printf(".....<-REAR\n");
   }
}
void main()
{
   int choice,i;
   do
   {
      printf("\n\n\n\n......MENU.........\n");
      printf("1. Insert an element to queue \n");
      printf("2. Delete an element from queue \n");
      printf("3. Display all elements of queue \n");
      printf("4. size to queue\n");
      printf("5. Quit \n");
      printf("Enter your choice : \n");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1: enqueue();display();
                 break;
         case 2: dequeue();shift();display();
                 break;
         case 3: display();
                 break;
         case 4: i=size();
                printf("the size of queue is : %d",i);
                break;
         case 5: printf("THANKYOU");
                 exit(0);
                break;
         default: printf("Wrong choice \n");
      }
   }while(choice!=5);
}
```

Riyanshi

Riyanshi



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\queue\priority_queue_3.exe
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
1

Enter the element to be inserted in queue :
12

Queue is :
FRONT->.....  45   78   12 .....<-REAR




......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. size to queue
5. Quit
Enter your choice :
2


The elements are--  12    45    78
Element deleted from queue is :  12

Queue is :
FRONT->.....  45   78 .....<-REAR
```

****

CIRCULAR QUEUE

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
#define MAX 5
int cqueue[MAX];
int front=-1;
int rear=-1;
void enqueue()
{
        int item;
        if((front==0 && rear== MAX-1)|| front==rear+1)
        {
                printf("\nQueue Overflow!!");
                return;
        }
        if(front==-1)
        {
                front=0;
                rear=0;
        }
        else if(rear==MAX-1)
        rear=0;
```

Riyanshi

Riyanshi

1913075

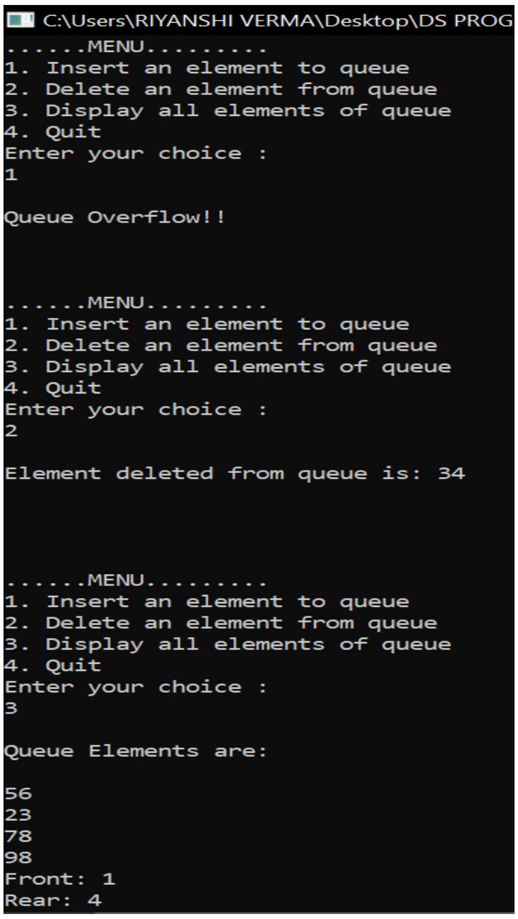
```
        else
        rear=rear+1;
        printf("\nInput the element for insertion in queue:\n");
        scanf("%d",&item);
        cqueue[rear]=item;
}
void dequeue()
{
        if(front==-1)
        {
                printf("\nQueue Underflow");
                return;
        }
        printf("\nElement deleted from queue is: %d\n",cqueue[front]);
        if(front ==rear)
        {
                front=-1;
                rear=-1;
        }
        else if(front==MAX-1)
        front=0;
        else
        front=front+1;
}
void display()
{
        int i;
        if(front==-1)
        {
                printf("\nQueue is empty\n");
                return;
        }
        printf("\nQueue Elements are:\n");
        if(rear>=front)
        {
                for(i=front;i<=rear;i++)
                printf("\n%d",cqueue[i]);
        }
        else
        {
                for(i=front;i<=MAX-1;i++)
                printf("\n%d",cqueue[i]);
                for(i=0;i<=rear;i++)
                printf("\n%d",cqueue[i]);
        }
        printf("\nFront: %d",front);
        printf("\nRear: %d",rear);

}
void main()
{
        int choice,i;
```

Riyanshi

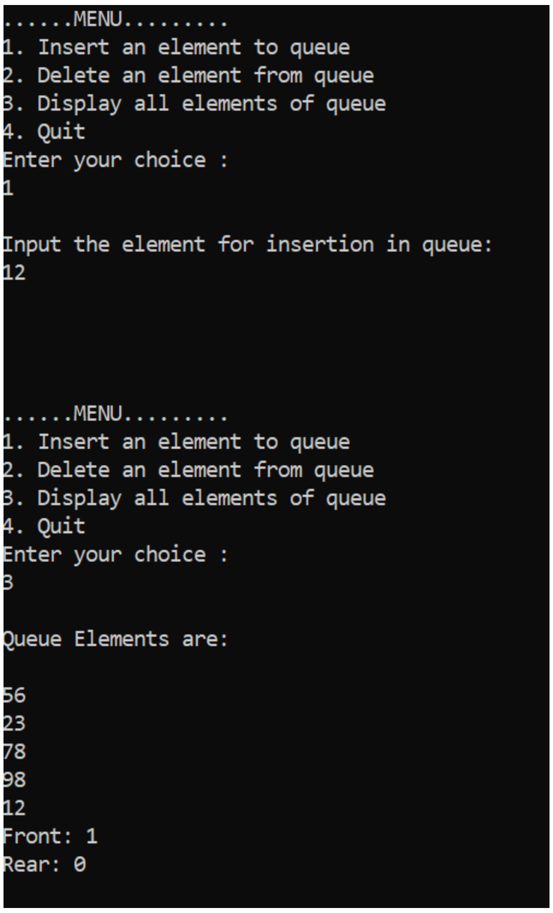
Riyanshi

1913075

```
  do
  {
     printf("\n\n\n\n......MENU.........\n");
     printf("1. Insert an element to queue \n");
     printf("2. Delete an element from queue \n");
     printf("3. Display all elements of queue \n");
     printf("4. Quit \n");
     printf("Enter your choice : \n");
     scanf("%d", &choice);
     switch (choice)
     {
        case 1: enqueue();
              break;
        case 2: dequeue();
              break;
        case 3: display();
              break;
        case 4: printf("THANKYOU");
                                   exit(0);
                                               break;
        default: printf("Wrong choice \n");
     }
  }while(choice!=5);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROG
......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
1

Queue Overflow!!



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
2

Element deleted from queue is: 34



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
3

Queue Elements are:

56
23
78
98
Front: 1
Rear: 4
```



```
......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
1

Input the element for insertion in queue:
12



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
3

Queue Elements are:

56
23
78
98
12
Front: 1
Rear: 0
```

Riyanshi

Riyanshi

1913075

****

CIRCULAR PRIORITY QUEUE

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
#define MAX 5
int cqueue[MAX];
int front=-1;
int rear=-1;
void sort()
{
            int j,x,i;
            if(rear>=front)
            {
                        for(i=front+1;i<=rear;i++)
            {
                           j=i;x=cqueue[i];
                           while(cqueue[j-1]>x && j>0)
                             {
                                       cqueue[j]=cqueue[j-1];
                                       j=j-1;
                             }
                           if (j!=i)
                           cqueue[j]=x;
               }
   printf("\n\nThe sorted list of elements is--  ");
            for(i=front;i<=rear;i++)
            {
                        printf("%d    ",cqueue[i]);
            }
            }
            else
            {
                        int r=rear,f=front;i=f+1;
                        do
                        {
                                  j=i;x=cqueue[i];
                           while(cqueue[j-1]>x && j>0)
                             {
                                       cqueue[j]=cqueue[j-1];
                                       j=j-1;
                             }
                           if (j!=i)
                           cqueue[j]=x;
                           if(i<=MAX-1)
                           i++;
                           else
                           {
```

Riyanshi

Riyanshi

```
                        i=(r+1)%MAX;
                    r++;
                        }

                  printf("r=%d f=%d",r,f);

                }while(i!=rear);
        for(i=front;i<=MAX-1;i++)
                printf("\n%d",cqueue[i]);
                for(i=0;i<=rear;i++)
                printf("\n%d",cqueue[i]);
        }
 }
void enqueue()
{
        int item;
        if((front==0 && rear== MAX-1)|| front==rear+1)
        {
                printf("\nQueue Overflow!!");
                return;
        }
        if(front==-1)
        {
                front=0;
                rear=0;
        }
        else if(rear==MAX-1)
        rear=0;
        else
        rear=rear+1;
        printf("\nInput the element for insertion in queue:\n");
        scanf("%d",&item);
        cqueue[rear]=item;
        sort();
}
void dequeue()
{
        if(front==-1)
        {
                printf("\nQueue Underflow");
                return;
        }

        printf("\nElement deleted from queue is: %d\n",cqueue[front]);

        if(front ==rear)
        {
                front=-1;
                rear=-1;
        }
        else if(front==MAX-1)
        front=0;
```

Riyanshi

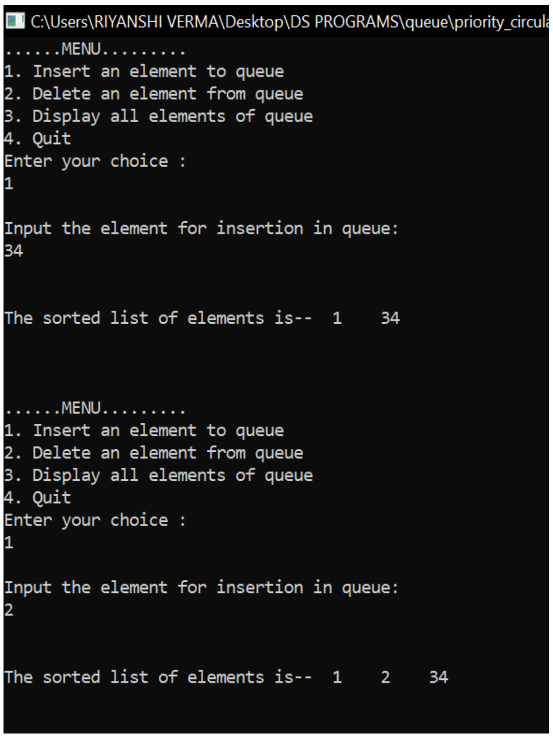
Riyanshi

```
            else
            front=front+1;
}
void display()
{
            int i;
            if(front==-1)
            {
                        printf("\nQueue is empty\n");
                        return;
            }
            printf("\nQueue Elements are:\n");
            if(rear>=front)
            {
                        for(i=front;i<=rear;i++)
                        printf("\n%d",cqueue[i]);
            }
            else
            {
                        for(i=front;i<=MAX-1;i++)
                        printf("\n%d",cqueue[i]);
                        for(i=0;i<=rear;i++)
                        printf("\n%d",cqueue[i]);
            }
            printf("\nFront: %d",front);
            printf("\nRear: %d",rear);

}
void main()
{
            int choice,i;

   do
   {
      printf("\n\n\n\n......MENU.........\n");
      printf("1. Insert an element to queue \n");
      printf("2. Delete an element from queue \n");
      printf("3. Display all elements of queue \n");
      printf("4. Quit \n");
      printf("Enter your choice : \n");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1: enqueue();
               break;
         case 2: dequeue();
               break;
         case 3: display();
               break;
         case 4: printf("THANKYOU");
                                exit(0);
                                            break;
```

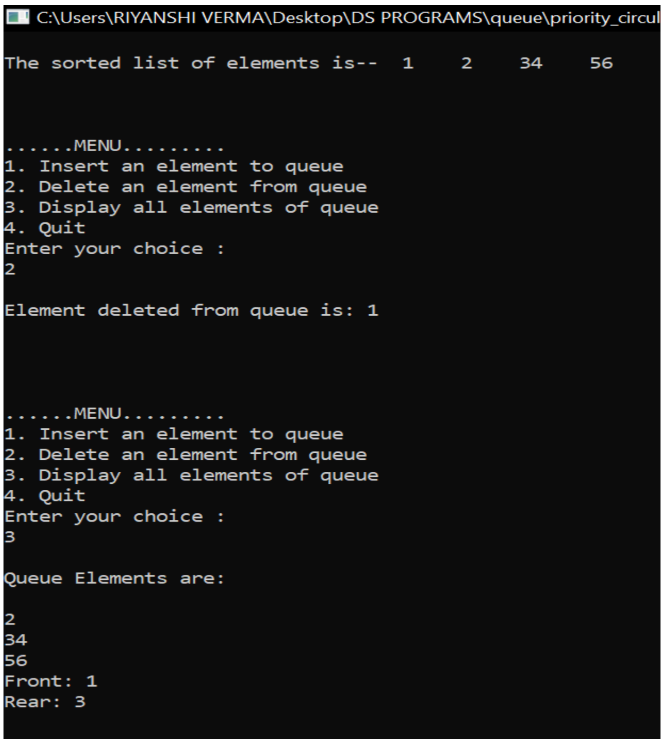
Riyanshi

Riyanshi

```
        default: printf("Wrong choice \n");
      }
   }while(choice!=5);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\queue\priority_circula
......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
1

Input the element for insertion in queue:
34


The sorted list of elements is--  1    34



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
1

Input the element for insertion in queue:
2


The sorted list of elements is--  1    2    34
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\queue\priority_circul

The sorted list of elements is--   1    2    34    56



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
2

Element deleted from queue is: 1



......MENU.........
1. Insert an element to queue
2. Delete an element from queue
3. Display all elements of queue
4. Quit
Enter your choice :
3

Queue Elements are:

2
34
56
Front:  1
Rear:  3
```

****

DEQUEUE

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node
{
   struct node *prev;
   struct node *next;
   int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void deletion_beginning();
void deletion_last();
void display();
void search();
void main ()
{
int choice =0;
   while(choice != 9)
   {
      printf("\nChoose one option from the following list ...\n");
```

Riyanshi

Riyanshi

```
        printf("\n1.Insert in Beginning\n2.Insert at last\n3.Delete from Beginning\n4.Delete from
last\n5.Search\n6.Show\n7.Exit\n");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
          case 1:
          insertion_beginning();
          break;
          case 2:
                insertion_last();
          break;
          case 3:
          deletion_beginning();
          break;
          case 4:
          deletion_last();
          break;
          case 5:
          search();
          break;
          case 6:
          display();
          break;
          case 7:
          exit(0);
          break;
          default:
          printf("Please enter valid choice..");
        }
      }
}
void insertion_beginning()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW");
   }
   else
   {
    printf("\nEnter Item value");
    scanf("%d",&item);
    ptr->data=item;
   if(head==NULL)
   {
     head = ptr;
     ptr -> next = head;
     ptr -> prev = head;
   }
```

Riyanshi

Riyanshi

```
  else
  {
     temp = head;
   while(temp -> next != head)
   {
      temp = temp -> next;
   }
   temp -> next = ptr;
   ptr -> prev = temp;
   head -> prev = ptr;
   ptr -> next = head;
   head = ptr;
  }
  printf("\nNode inserted\n");
}

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

Riyanshi

Riyanshi

```
void deletion_beginning()
{
   struct node *temp;
   if(head == NULL)
   {
      printf("\n UNDERFLOW");
   }
   else if(head->next == head)
   {
      head = NULL;
      free(head);
      printf("\nnode deleted\n");
   }
   else
   {
      temp = head;
      while(temp -> next != head)
      {
         temp = temp -> next;
      }
      temp -> next = head -> next;
      head -> next -> prev = temp;
      free(head);
      head = temp -> next;
   }

}
void deletion_last()
{
   struct node *ptr;
   if(head == NULL)
   {
      printf("\n UNDERFLOW");
   }
   else if(head->next == head)
   {
      head = NULL;
      free(head);
      printf("\nnode deleted\n");
   }
   else
   {
      ptr = head;
      if(ptr->next != head)
      {
         ptr = ptr -> next;
      }
      ptr -> prev -> next = head;
      head -> prev = ptr -> prev;
      free(ptr);
      printf("\nnode deleted\n");
```

Riyanshi

Riyanshi

```
    }
}

void display()
{
    struct node *ptr;
    ptr=head;
    if(head == NULL)
    {
        printf("\nnothing to print");
    }
    else
    {
        printf("\n printing values ... \n");

        while(ptr -> next != head)
        {

            printf("%d\n", ptr -> data);
            ptr = ptr -> next;
        }
        printf("%d\n", ptr -> data);
    }

}

void search()
{
    struct node *ptr;
    int item,i=0,flag=1;
    ptr = head;
    if(ptr == NULL)
    {
        printf("\nEmpty List\n");
    }
    else
    {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        if(head ->data == item)
        {
        printf("item found at location %d",i+1);
        flag=0;
        }
        else
        {
        while (ptr->next != head)
        {
            if(ptr->data == item)
            {
                printf("item found at location %d ",i+1);
                flag=0;
```

Riyanshi

Riyanshi

```
            break;
          }
          else
          {
            flag=1;
          }
          i++;
          ptr = ptr -> next;
        }
        }
        if(flag != 0)
        {
          printf("Item not found\n");
        }
      }

    }
```
****

QUICK SORT

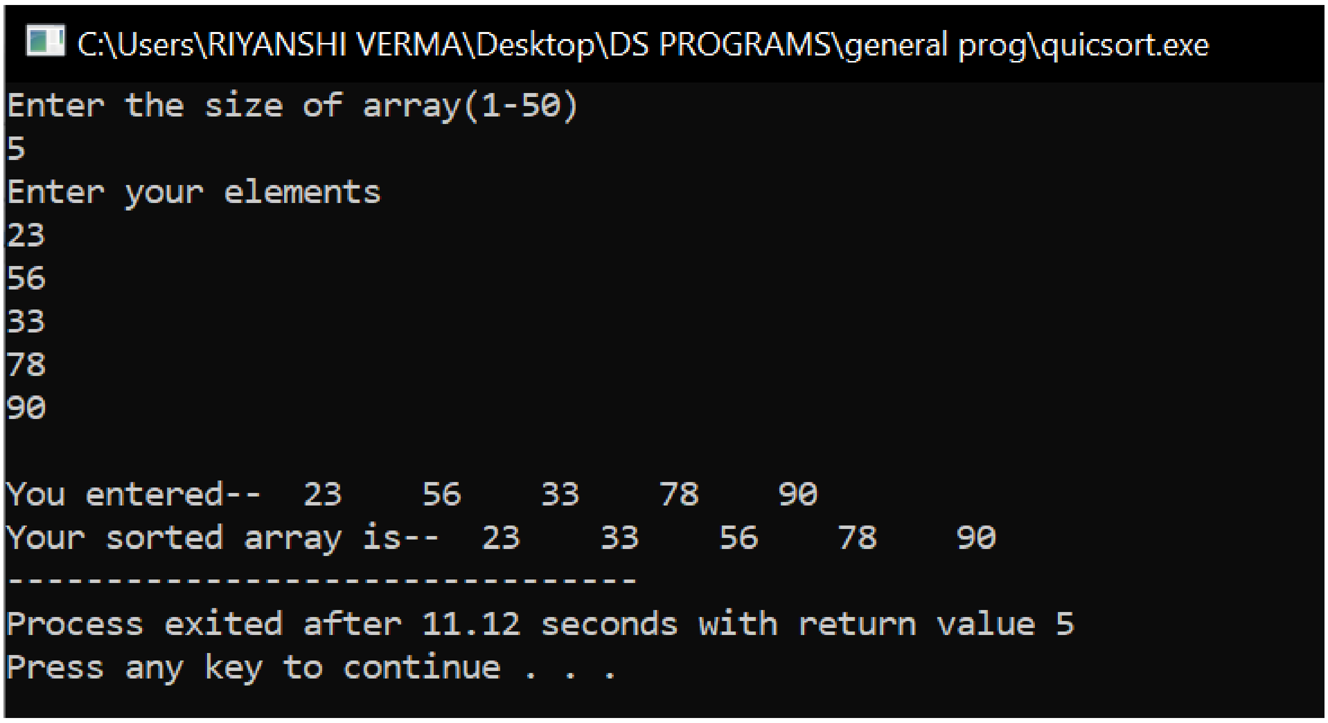
```
#include<stdio.h>
#include<stdlib.h>
#define maxarray 50
void quicksort(int arr[],int low,int high);
void main()
{
	int n;
	printf("Enter the size of array(1-50)\n");
	scanf("%d",&n);
	if(n<0 || n>50)
	{
		printf("Enter correct value of n\n ");
		scanf("%d",&n);
	}
	if(n<0 || n>50)
	{
		exit(0);
	}
	int array[50];
	int i;
	printf("Enter your elements\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&array[i]);
	}
	printf("\nYou entered--  ");
	for(i=0;i<n;i++)
	{
		printf("%d    ",array[i]);
	}
  quicksort(array,0,(n-1));
	printf("\nYour sorted array is--  ");
```

Riyanshi

Riyanshi

1913075

```
		for(i=0;i<n;i++)
		{
			printf("%d    ",array[i]);
		}
}
void quicksort(int arr[],int low,int high)
{
		int i=low;//l
		int j=high;//r
		int y=0,k;
		int z=arr[(low+high)/2];//z=low
		do{
			while(arr[i]<z)
			i++;
			while(arr[j]>z)
			j--;
			if(i<=j)
			{
				y=arr[i];
				arr[i]=arr[j];
				arr[j]=y;
				i++;j--;
			}
		}while(i<=j); //swap z and arr[j]
		if(low<j)
		quicksort(arr,low,j);
		if(i<high)
		quicksort(arr,i,high);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\quicsort.exe
Enter the size of array(1-50)
5
Enter your elements
23
56
33
78
90

You entered--  23    56    33    78    90
Your sorted array is--  23    33    56    78    90
--------------------------------
Process exited after 11.12 seconds with return value 5
Press any key to continue . . .
```

****

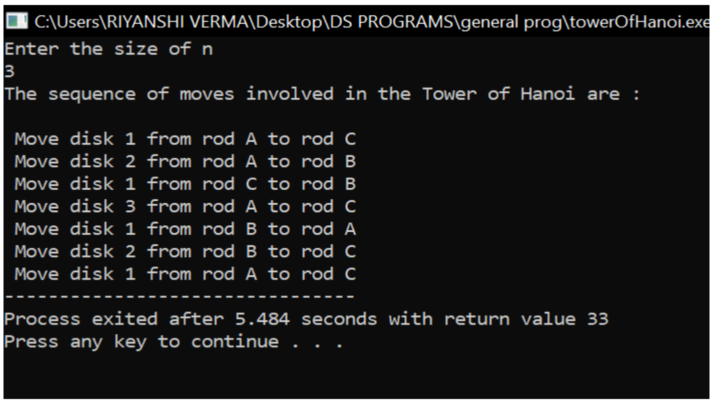
Riyanshi

Riyanshi

1913075

TOWER OF HANOI

```
#include <stdio.h>

void ToH(int n, char a, char c, char b)
{
   if (n == 1)
   {
      printf("\n Move disk 1 from rod %c to rod %c", a,c);
      return;
   }
   ToH(n-1,a,b,c);
   printf("\n Move disk %d from rod %c to rod %c", n, a,c);
   ToH(n-1,b,c,a);
}

void main()
{
   int n;
   printf("Enter the size of n\n");
   scanf("%d",&n);
         printf("The sequence of moves involved in the Tower of Hanoi are :\n");
   ToH(n, 'A', 'C', 'B');

}
```



C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\general prog\towerOfHanoi.exe

```
Enter the size of n
3
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from rod A to rod C
 Move disk 2 from rod A to rod B
 Move disk 1 from rod C to rod B
 Move disk 3 from rod A to rod C
 Move disk 1 from rod B to rod A
 Move disk 2 from rod B to rod C
 Move disk 1 from rod A to rod C
--------------------------------
Process exited after 5.484 seconds with return value 33
Press any key to continue . . .
```

SINGLE LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
         int data;
         struct node *next;
}*head;
void move_to_first(int a)
{
   struct node *prv=head,*cur=head;
```

Riyanshi

Riyanshi

1913075

```
    while(cur!=NULL)
            {
       if(a==cur->data)
                      {
          //Found the item
          prv->next=cur->next;
          cur->next=head;
          head=cur;
          return;
       }
       prv = cur;
       cur=cur->next;
    }
    printf("\nElement not found\n");
}
void swap(int x, int y)
{
    struct node *prv1, *prv2, *node1 = head, *node2 = head, *temp;


    if(head == NULL) {
            printf("\nLinked list is empty\n");
       return;
    }

    if(x == y)
       return;
    while(node1 != NULL && node1->data != x){
       prv1 = node1;
       node1 = node1->next;
    }

    while(node2 != NULL && node2->data != y){
       prv2 = node2;
       node2 = node2->next;
    }

    if(node1 != NULL && node2 != NULL) {

       if(prv1 != NULL)
          prv1->next = node2;
       else
          head  = node2;
       if(prv2 != NULL)
          prv2->next = node1;
       else
          head  = node1;



       temp = node1->next;
       node1->next = node2->next;
       node2->next = temp;
```

Riyanshi

Riyanshi

```
   }
   else{
      printf("Swapping is not possible\n");
   }
}
int count()
{
        int c=0;
        struct node *cur;
        cur=head;
        while(cur!=NULL)
        {
                cur=cur->next;
                c++;
        }
        return c;
}
void search(int num)
{
        struct node *cur;int i=0;
        cur=head;
        while(cur!=NULL)
        {
                i++;
                if(cur->data==num)
                {
                        printf("\nElement  %d found at %d",num, i);
                        return;
                }
                else
                cur=cur->next;
        }
        printf("\nElement %d not found",num);
}
void beginninginsert()
{
        int num;
        printf("Enter the number\n");
        scanf("%d",&num);
        struct node *newnode;
   newnode =(struct node *)malloc(sizeof(struct node));
   newnode->data=num;
        newnode->next=NULL;
        if(head==NULL)
        {
                head=newnode;
                head->next=NULL;
        }
        else
        {
                newnode->next=head;
                head=newnode;
```

Riyanshi

Riyanshi

```
            }
}
void lastinsert()
{
            int num;
            printf("\n\nEnter the number\n");
            scanf("%d",&num);
            struct node *newnode;
   newnode =(struct node *)malloc(sizeof(struct node));
            newnode->data=num;
            newnode->next=NULL;
            if(head==NULL)
            {
                        head=newnode;
                        head->next=NULL;
            }
            else
            {
                        struct node *cur;
                        cur=head;
                        while(cur->next!=NULL)
                        cur=cur->next;
                        cur->next=newnode;
            }
}
void insertloc(int loc)
{
            int i;
            struct node *newnode,*prv,*cur;
            cur=head;
            if(loc>count()+1 || loc <=0)
            {
                        printf("\nInsertion not possible");
                        return;
            }
            if(loc==1)
            {
                        beginninginsert();
            }
            else
            {
                        for(i=1;i<loc;i++)
                        {
                                    prv=cur;
                                    cur=cur->next;
                        }
                                    int num;
            printf("\n\nEnter the number\n");
            scanf("%d",&num);
   newnode =(struct node *)malloc(sizeof(struct node));
            newnode->data=num;
            prv->next=newnode;
```

Riyanshi

Riyanshi

```
        prv=newnode;
        prv->next=cur;
        return;
        }
}
void deletefirst()
{
        if(head==NULL)
        {
                printf("\nNO element found");
                return;
        }

        else
        {
                struct node *cur;
                cur=head;
                head=cur->next;
                free(cur);
        }
}
void deletelast()
{
        struct node *cur, *prv;
        if(head==NULL)
        {
                printf("\nLinkedlist underflow");
                return;
        }

        else
        {

                cur=head;
                while(cur->next!=NULL)
                {
                        prv=cur;
                        cur=cur->next;
                }
        }
        prv->next=NULL;
        free(cur);
}
void delectloc(int loc)
{
        int i;
        struct node *newnode,*prv,*cur;
        cur=head;
        if(loc>count()+1 || loc <=0)
        {
                printf("\nDeletion not possible");
                return;
```

Riyanshi

Riyanshi

1913075

```
	}
	if(loc==1)
	{
		deletefirst();
	}
	else
	{
		for(i=1;i<loc;i++)
		{
			prv=cur;
			cur=cur->next;
		}
		prv->next=cur->next;
		free(cur);
		return;
}
}

void printing()
{
	struct node *cur;
		cur=head;
		while(cur!=NULL)
		{
			printf("%d->",cur->data);
			cur=cur->next;
		}
}
void main()
{
	head=NULL;int t,l,n,c;int a,b;
	int choice;
	do
	{
		printf("\nHERE IS THE CHOICE\n1:INSERT A NODE \n2:SEARCH A NODE AT LAST\n3.COUNT
THE NUMBER OF NODES\n4.DELETE A NODE \n5:DISPLAY THE NODE\n6:SWAP THE NODES\n7:MOVE THE
NODE TO FIRST\n8:QUIT");
		printf("\nEnter the choice:\n");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:printf("\n1.INSERT AT BEGINNING\n2.INSERT AT LAST\n3.INSERT AT
PARTICULAR LOCATION");
				printf("\nEnter your choice:\n");
				scanf("%d",&t);
				switch(t)
				{
					case 1:beginninginsert();break;
					case 2:lastinsert();break;
					case 3:printf("\nEnter the location\n");
						scanf("%d",&l);
						insertloc(l);
```

Riyanshi

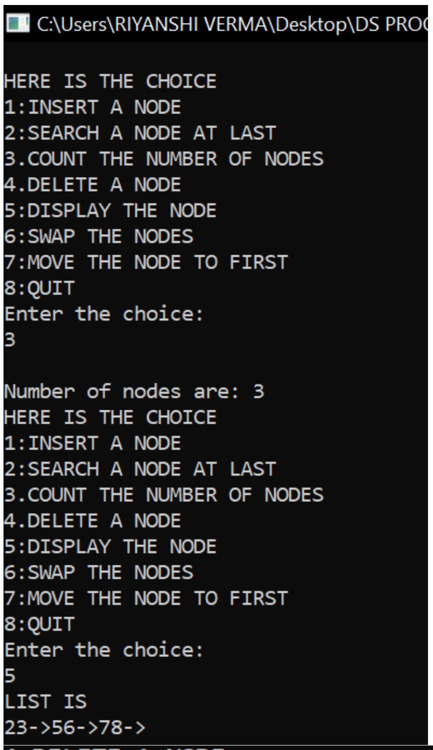
Riyanshi

1913075

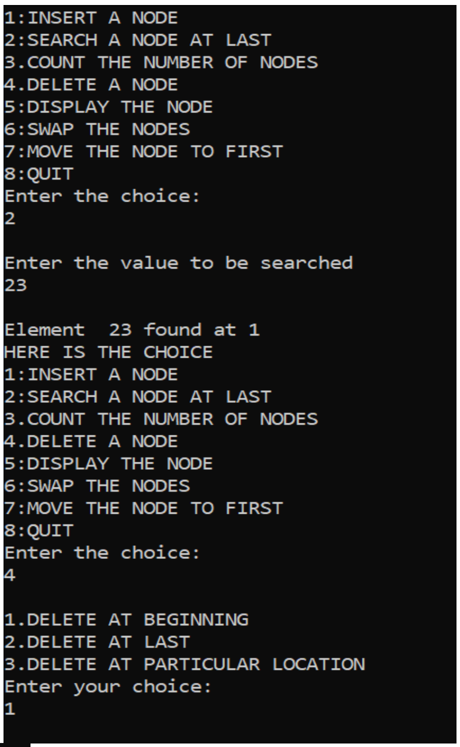
```
                                        break;
                          }
                         break;
                       case 2: printf("\nEnter the value to be searched\n");
                              scanf("%d",&n);
                              search(n);
                              break;
                       case 3: c=count();
                              printf("\nNumber of nodes are: %d",c);
                               break;
                       case 4:printf("\n1.DELETE AT BEGINNING\n2.DELETE AT LAST\n3.DELETE AT
PARTICULAR LOCATION");
                          printf("\nEnter your choice:\n");
                          scanf("%d",&t);
                          switch(t)
                          {
                                    case 1:deletefirst();break;
                                    case 2:deletelast();break;
                                    case 3:printf("\nEnter the location\n");
                                        scanf("%d",&l);
                                        delectloc(l);
                                        break;
                               }
                               break;
                       case 5: printf("LIST IS\n");
                              printing();
                              break;
                       case 6: printf("\nEnter the swapping values\n");
                              scanf("%d",&a);
                              scanf("%d",&b);
                              swap(a,b);
                              printing();
                              break;
                       case 7: printf("\nEnter the node to be moved\n");
                              scanf("%d",&a);
                              move_to_first(a);
                              printing();
                              break;
                       case 8: exit(0);
                              default:
                              printf("**********************\nWRONG
CHOICE\n**********************\n");

           }

      }while(choice!=8);
}
```

Riyanshi

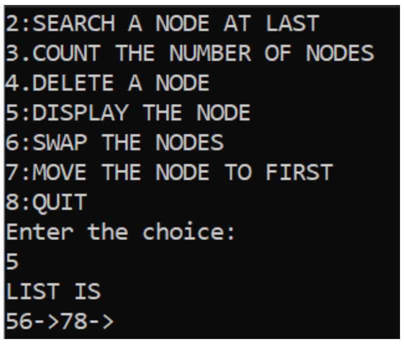
Riyanshi



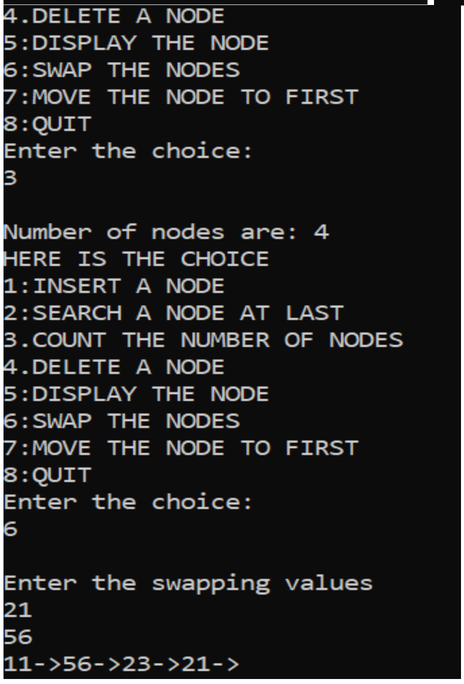
```
C:\Users\RIYANSHI VERMA\Desktop\DS PROC

HERE IS THE CHOICE
1:INSERT A NODE
2:SEARCH A NODE AT LAST
3.COUNT THE NUMBER OF NODES
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
3

Number of nodes are: 3
HERE IS THE CHOICE
1:INSERT A NODE
2:SEARCH A NODE AT LAST
3.COUNT THE NUMBER OF NODES
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
5
LIST IS
23->56->78->
```



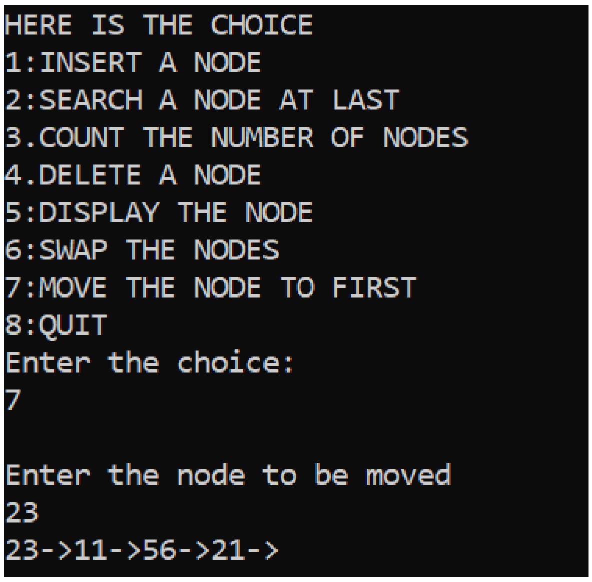
```
1:INSERT A NODE
2:SEARCH A NODE AT LAST
3.COUNT THE NUMBER OF NODES
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
2

Enter the value to be searched
23

Element  23 found at 1
HERE IS THE CHOICE
1:INSERT A NODE
2:SEARCH A NODE AT LAST
3.COUNT THE NUMBER OF NODES
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
4

1.DELETE AT BEGINNING
2.DELETE AT LAST
3.DELETE AT PARTICULAR LOCATION
Enter your choice:
1
```



```
2:SEARCH A NODE AT LAST
3.COUNT THE NUMBER OF NODES
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
5
LIST IS
56->78->
```



```
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
3

Number of nodes are: 4
HERE IS THE CHOICE
1:INSERT A NODE
2:SEARCH A NODE AT LAST
3.COUNT THE NUMBER OF NODES
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
6

Enter the swapping values
21
56
11->56->23->21->
```



```
HERE IS THE CHOICE
1:INSERT A NODE
2:SEARCH A NODE AT LAST
3.COUNT THE NUMBER OF NODES
4.DELETE A NODE
5:DISPLAY THE NODE
6:SWAP THE NODES
7:MOVE THE NODE TO FIRST
8:QUIT
Enter the choice:
7

Enter the node to be moved
23
23->11->56->21->
```

****

<u>STACK USING LINKED LIST</u>

```
#include <stdio.h>
#include <stdlib.h>
#include<process.h>
struct node
{
int data;
struct node *next;
}*head;
```

Riyanshi

Riyanshi

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

void pop()
{
   int item;
   struct node *temp;
   if (top == NULL)
   {
      printf("Underflow");
                  return;
   }
   else
   {
      item = top->data;
                  printf("\nThe value returned is %d",item);
      temp = top;
      top = top->next;
      free(temp);
   }
}
void display()
{
   int i;
   struct node *ptr;
   ptr=top;
   if(top == NULL)
   {
      printf("\nStack is empty\n");
                  return;
   }
   else
   {
      printf("\nSTACK elements: \n");
      while(ptr!=NULL)
      {
         printf("%d\n",ptr->data);
         ptr = ptr->next;
      }
   }
}
```

Riyanshi

Riyanshi

```
void main ()
{

   top=NULL;
   int choice;
            do
            {
      printf("\n\n\nStack operations using linked list");
      printf("\n1.Push an element\n2.Pop an element\n3.Display\n4.Exit");
      printf("\nEnter your choice \n");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1: push(); display();
               break;
         case 2: pop(); display();
               break;
         case 3: display();
               break;
         case 4: printf("\nTHANKYOU......");
                                   exit(0);
               break;
         default:printf("\nWRONG CHOICE");
      }
  }while(choice!=4);
}
```

OUTPUT SAME AS IN STACK

****

QUEUE USING LINKED LIST

```
#include <stdio.h>
#include <stdlib.h>
#include<process.h>
struct node
{
            int data;
            struct node* next;
};
struct node* rear, *front;
int dequeue()
{
            if (front == NULL)
            {
                       printf("\nQueue Underflow");
                       return;
            }
   struct node *temp = front;
            printf("Removing the value %d\n", temp->data);
            front = front->next;
            if (front == NULL)
```

Riyanshi

Riyanshi

```
                rear = NULL;
        int item = temp->data;
        free(temp);
        return item;
}

void enqueue()
{
        struct node *newnode;
        newnode = (struct node*)malloc(sizeof(struct node));
        printf("Enter the value\n");
        int item;
  scanf("%d",&item);
  newnode->data = item;
  newnode-> next = NULL;
        printf("Inserting the value %d\n", item);

        if (front == NULL)
        {
                front = newnode;
                rear = newnode;
        }
        else
        {
                rear->next = newnode;
                rear = newnode;
        }
}
void display()
{
        struct node* temp;
        temp=front;
        if(temp==NULL)
        printf("\nQueue Underflow");
        else
        {
                printf("\nFront ");
                while(temp!=NULL)
                {
                        printf("%d->",temp->data);
                        temp=temp->next;
                }
                printf(" Rear");
        }
}

void main()
{
front =NULL, rear=NULL;
  int choice;
        do
        {
```

Riyanshi

Riyanshi

```
        printf("\n\n\nQueue operations using linked list");
        printf("\n1.Enqueue an element\n2.Dequeue an element\n3.Display\n4.Exit");
        printf("\nEnter your choice \n");
        scanf("%d",&choice);
        switch(choice)
        {
          case 1: enqueue(); display();
                break;
          case 2: dequeue(); display();
                break;
          case 3: display();
                break;
          case 4: printf("\nTHANKYOU......");
                                exit(0);
                break;
          default:printf("\nWRONG CHOICE");
        }
    }while(choice!=4);
}
```

OUTPUT SAME AS IN QUEUE

****

DOUBLY LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
        int data;
        struct node *next;
        struct node *prev;
}*head;
int count()
{
        int c=0;
        struct node *cur;
        cur=head;
        while(cur!=NULL)
        {
                cur=cur->next;
                c++;
        }
        return c;
}
void search(int num)
{
        struct node *cur;int i=0;
        cur=head;
        while(cur!=NULL)
        {
                i++;
```

Riyanshi

Riyanshi

```
			if(cur->data==num)
			{
				printf("\nElement  %d found at %d",num, i);
				return;
			}
			else
			cur=cur->next;
		}
		printf("\nElement %d not found",num);
}
void beginninginsert()
{
		int num;
		printf("Enter the number\n");
		scanf("%d",&num);
		struct node *newnode;
   newnode =(struct node *)malloc(sizeof(struct node));
   newnode->data=num;
		newnode->next=NULL;
		if(head==NULL)
		{
			head=newnode;
			head->next=NULL;
			head->prev=NULL;
		}
		else
		{
			newnode->next=head;
			head->prev=newnode;
			head=newnode;
		}
}
void lastinsert()
{
		int num;
		printf("\n\nEnter the number\n");
		scanf("%d",&num);
		struct node *newnode;
   newnode =(struct node *)malloc(sizeof(struct node));
		newnode->data=num;
		newnode->next=NULL;
		if(head==NULL)
		{
			head=newnode;
			head->next=NULL;
			head->prev=NULL;
		}
		else
		{
			struct node *cur;
			cur=head;
			while(cur->next!=NULL)
```

Riyanshi

Riyanshi

```
                    cur=cur->next;
                    cur->next=newnode;
                    newnode->prev=cur;
            }
}
void insertloc(int loc)
{
            int i;
            struct node *newnode,*prv,*cur;
            cur=head;
            if(loc>count()+1 || loc <=0)
            {
                    printf("\nInsertion not possible");
                    return;
            }
            if(loc==1)
            {
                    beginninginsert();
            }
            else
            {
                    for(i=1;i<loc;i++)
                    {
                            prv=cur;
                            cur=cur->next;
                    }
            int num;
            printf("\n\nEnter the number\n");
            scanf("%d",&num);
   newnode =(struct node *)malloc(sizeof(struct node));
            newnode->data=num;
            prv->next=newnode;
            newnode->next=cur;
            cur->prev=newnode;
            newnode->prev=prv;
            return;
            }
}
void deletefirst()
{
            if(head==NULL)
            {
                    printf("\nNO element found");
                    return;
            }

            else
            {
                    struct node *cur;
                    cur=head;
                    head=cur->next;
                    head->prev=NULL;
```

Riyanshi

Riyanshi

```
                    free(cur);
          }
}
void deletelast()
{
          struct node *cur, *prv;
          if(head==NULL)
          {
                    printf("\nLinkedlist underflow");
                    return;
          }

          else
          {

                    cur=head;
                    while(cur->next!=NULL)
                    {
                              prv=cur;
                              cur=cur->next;
                    }
          }
          cur->prev->next=NULL;
          free(cur);
}
void delectloc(int loc)
{
          int i;
          struct node *newnode,*prv,*cur;
          cur=head;
          if(loc>count()+1 || loc <=0)
          {
                    printf("\nDeletion not possible");
                    return;
          }
          if(loc==1)
          {
                    deletefirst();
          }
          else
          {
                    for(i=1;i<loc;i++)
                    {
                              cur=cur->next;
                    }
                    if(loc==count())
                    {
                              cur->prev->next=NULL;
                              free(cur);return;
                    }
                    cur->prev->next=cur->next;
                    cur->next->prev=cur->prev;
```

Riyanshi

Riyanshi

```
			free(cur);
			return;
	}
	}

void printingbackward()
{
		struct node *cur;
			cur=head;
			while(cur->next!=NULL)
			cur=cur->next;
			printf("\nBackward traversal=  ");
			while(cur!=NULL)
			{
				printf("%d<-",cur->data);
				cur=cur->prev;
			}

}
void printingforward()
{
		struct node *cur;
			cur=head;
			printf("\nForward traversal=  ");
			while(cur!=NULL)
			{
				printf("%d->",cur->data);
				cur=cur->next;
			}
}
void main()
{
		head=NULL;int t,l,n,c;int a,b;
		int choice;
		do
		{
			printf("\nHERE IS THE CHOICE\n1:INSERT A NODE \n2:SEARCH A NODE AT LAST\n3.COUNT
THE NUMBER OF NODES\n4.DELETE A NODE \n5:DISPLAY THE NODE\n6:QUIT");
			printf("\nEnter the choice:\n");
			scanf("%d",&choice);
			switch(choice)
			{
				case 1:printf("\n1.INSERT AT BEGINNING\n2.INSERT AT LAST\n3.INSERT AT
PARTICULAR LOCATION");
					printf("\nEnter your choice:\n");
					scanf("%d",&t);
					switch(t)
					{
					   case 1:beginninginsert();break;
						case 2:lastinsert();break;
						case 3:printf("\nEnter the location\n");
						   scanf("%d",&l);
```

Riyanshi

Riyanshi

```
                                        insertloc(l);
                                        break;
                                }
                                break;
                        case 2: printf("\nEnter the value to be searched\n");
                                scanf("%d",&n);
                        search(n);
                                break;
                        case 3: c=count();
                                printf("\nNumber of nodes are: %d",c);
                                break;
                        case 4:printf("\n1.DELETE AT BEGINNING\n2.DELETE AT LAST\n3.DELETE AT
PARTICULAR LOCATION");
                                printf("\nEnter your choice:\n");
                                scanf("%d",&t);
                                switch(t)
                                {
                                    case 1:deletefirst();break;
                                        case 2:deletelast();break;
                                        case 3:printf("\nEnter the location\n");
                                            scanf("%d",&l);
                                            delectloc(l);
                                            break;
                                }
                                break;
                        case 5: printf("LIST IS\n");
                                printingforward();
                                printingbackward();
                                break;

                    case 6: exit(0);
                        default:
                                printf("*********************\nWRONG
CHOICE\n*********************\n");

                }

        }while(choice!=6);
}
```

OUTPUT SAME AS LL

****

CIRCULAR LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
        int data;
```

Riyanshi

Riyanshi

1913075

```
            struct node *next;
}*head;
int count()
{
            int c=0;
            struct node *cur;
            if(head!=NULL)
            {
            cur=head;
            while(cur->next!=head)
            {
                        cur=cur->next;
                        c++;
            }
            c=c+1;
            }
return c;
}
void search(int num)
{
            struct node *cur;int i=0;
            cur=head;
            while(cur->next!=head)
            {
                        i++;
                        if(cur->data==num)
                        {
                                    printf("\nElement  %d found at %d",num, i);
                                    return;
                        }
                        cur=cur->next;
            }
            i++;
            if(cur->next==head && cur->data==num)
            {
                        printf("\nElement  %d found at %d",num,i);
                        return;
            }
            printf("\nElement %d not found",num);
}
void beginninginsert()
{
            int num;
            printf("Enter the number\n");
            scanf("%d",&num);
            struct node *newnode,*cur;
   newnode =(struct node *)malloc(sizeof(struct node));
   newnode->data=num;
            newnode->next=NULL;
            if(head==NULL)
            {
                        head=newnode;
```

Riyanshi

Riyanshi

1913075

```
                head->next=head;
        }
        else
        {
                cur=head;
                newnode->next=head;
                while(cur->next!=head)
                        cur=cur->next;
                        cur->next=newnode;
                        head=newnode;
        }
}
void lastinsert()
{
        int num;
        printf("\n\nEnter the number\n");
        scanf("%d",&num);
        struct node *newnode;
  newnode =(struct node *)malloc(sizeof(struct node));
        newnode->data=num;

        if(head==NULL)
        {
                head=newnode;
                head->next=head;
        }
        else
        {
                struct node *cur;
                cur=head;
                while(cur->next!=head)
                cur=cur->next;
                cur->next=newnode;
                newnode->next=head;
        }
}
void insertloc(int loc)
{
        int i;
        struct node *newnode,*prv,*cur;
        cur=head;
        if(loc>count()+1 || loc <=0)
        {
                printf("\nInsertion not possible");
                return;
        }
        if(loc==1)
        {
                beginninginsert();
        }
        else
        {
```

Riyanshi

Riyanshi

```
                    for(i=1;i<loc;i++)
                    {
                              prv=cur;
                              cur=cur->next;
                    }
          int num;
          printf("\n\nEnter the number\n");
          scanf("%d",&num);
   newnode =(struct node *)malloc(sizeof(struct node));
          newnode->data=num;
          prv->next=newnode;
          prv=newnode;
          prv->next=cur;
          return;
          }
}
void deletefirst()
{
          if(head==NULL)
          {
                    printf("\nNO element found");
                    return;
          }

          else
          {
                    struct node *cur,*temp;
                    temp=head;
                    cur=head;
                    while(cur->next!=head)
                              cur=cur->next;
                    head=head->next;
                    cur->next=head;

          }
}
void deletelast()
{
          struct node *cur, *prv;
          if(head==NULL)
          {
                    printf("\nLinkedlist underflow");
                    return;
          }

          else
          {


                    cur=head;
                    while(cur->next!=head)
                    {
                              prv=cur;
```

Riyanshi

Riyanshi

```
                        cur=cur->next;
                }
        }
        prv->next=head;
        free(cur);
}
void delectloc(int loc)
{
        int i;
        struct node *newnode,*prv,*cur;
        cur=head;
        if(loc>count()+1 || loc <=0)
        {
                printf("\nDeletion not possible");
                return;
        }
        if(loc==1)
        {
                deletefirst();
        }
        if(loc==count())
        {
                deletelast();
        }
        else
        {
                for(i=1;i<loc;i++)
                {
                        prv=cur;
                        cur=cur->next;
                }
                prv->next=cur->next;
                free(cur);
                return;
}
}

void printing()
{
        struct node *cur;
                cur=head;
                while(cur->next!=head)
                {
                        printf("%d->",cur->data);
                        cur=cur->next;
                }
                printf("%d ",cur->data);
}
void main()
{
        head=NULL;int t,l,n,c;int a,b;
        int choice;
```

Riyanshi

Riyanshi

```
        do
        {
                printf("\nHERE IS THE CHOICE\n1:INSERT A NODE \n2:SEARCH A NODE IN THE LIST\n3.COUNT
THE NUMBER OF NODES\n4.DELETE A NODE \n5:DISPLAY THE NODE\n6:QUIT");
                printf("\nEnter the choice:\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:printf("\n1.INSERT AT BEGINNING\n2.INSERT AT LAST\n3.INSERT AT
PARTICULAR LOCATION");
                          printf("\nEnter your choice:\n");
                          scanf("%d",&t);
                          switch(t)
                          {
                             case 1:beginninginsert();break;
                             case 2:lastinsert();break;
                             case 3:printf("\nEnter the location\n");
                                  scanf("%d",&l);
                                  insertloc(l);
                                  break;
                              }
                              break;
                        case 2: printf("\nEnter the value to be searched\n");
                             scanf("%d",&n);
                             search(n);
                             break;
                        case 3: c=count();
                             printf("\nNumber of nodes are: %d",c);
                             break;
                        case 4:printf("\n1.DELETE AT BEGINNING\n2.DELETE AT LAST\n3.DELETE AT
PARTICULAR LOCATION");
                          printf("\nEnter your choice:\n");
                          scanf("%d",&t);
                          switch(t)
                          {
                             case 1:deletefirst();break;
                             case 2:deletelast();break;
                             case 3:printf("\nEnter the location\n");
                                  scanf("%d",&l);
                                  delectloc(l);
                                  break;
                              }
                              break;
                        case 5: printf("LIST IS\n");
                           printing();
                           break;
                    case 6: exit(0);
                        default:
                             printf("**********************\nWRONG
CHOICE\n**********************\n");

                }
```

Riyanshi

Riyanshi

```
          }while(choice!=8);
}
```

OUTPUT SAME AS LL

****

DOUBLY CIRCULAR LINKED LIST

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
          int data;
          struct node *next;
          struct node *prev;
}*head;
struct node *head = NULL, *tail = NULL;

void insert_at_head();
void insert_at_tail();
void insert_at_middle( int loc);
void delete_head();
void delete_tail();
void delete_middle(int loc);
void print_forward_order();
void print_reverse_order();
void print_list();
int getListLength();
void insert_at_head()
{
          int num;
          printf("Enter the number\n");
          scanf("%d",&num);
          struct node *newnode;
   newnode =(struct node *)malloc(sizeof(struct node));

  newnode->data= num;
  newnode->next = newnode;
  newnode->prev = newnode;

  if(head==NULL)
  {
     head = newnode;
     tail = newnode;
  }
  else
  {
     newnode->next = head;
     newnode->prev = tail;
     head->prev = newnode;
     tail->next = newnode;
     head = newnode;
```

Riyanshi

Riyanshi

```
    }

}

void insert_at_tail()
{
            int num;
            printf("Enter the number\n");
            scanf("%d",&num);
            struct node *newnode;
     newnode =(struct node *)malloc(sizeof(struct node));

    newnode->data = num;
    newnode->next = newnode;
    newnode->prev = newnode;

    if(head==NULL)
    {
        head = newnode;
        tail = newnode;
    }
    else
    {
        tail->next = newnode;
        newnode->next = head;
        newnode->prev = tail;
        tail = newnode;
        head->prev = tail;
    }
}

void insert_at_middle( int loc)
{
    if(loc==1)
    {
        insert_at_head();
        return;
    }
    else if(loc>1 && head!=NULL)
    {
        struct node *current = head;
        struct node *temp;
        temp =(struct node *)malloc(sizeof(struct node));
        int count = 0;

        do
        {
            count++;
            temp = current;
            current = current->next;
        }   while(current->next != head && count<loc-1);
```

Riyanshi

Riyanshi

```
        if(count==loc-1)
        {
           if(temp==tail)
              insert_at_tail();
           else
           {
             int num;
                     printf("Enter the number\n");
                     scanf("%d",&num);
                     struct node *newnode;
               newnode =(struct node *)malloc(sizeof(struct node));
              newnode->data = num;

              temp->next = newnode;
              newnode->next = current;
              newnode->prev = temp;
              current->prev = newnode;

           }
           return;
        }

    }

    printf("Position does not exist!\n");
}

void delete_head()
{
    if(head==NULL)
    return;

    struct node *temp = head;

    tail->next = head->next;
    head = head->next;
    head->prev = tail;

    free(temp);
}

void delete_tail()
{
    if(head==NULL)
            return;

    struct node *temp = head;
    struct node *current = head;

    while(current->next != head)
    {
       temp = current;
```

Riyanshi

Riyanshi

```
        current = current->next;
    }

    temp->next = head;
    tail = temp;
    head->prev = tail;
    free(current);
}


void delete_middle(int loc)
{
    if(head==NULL)
            return;

    if(loc==1)
    {
        delete_head();
        return;
    }

    struct node *current = head;
    struct node *temp;
    int count = 0;

    do
    {
        count++;
        temp = current;
        current = current->next;
    }   while(current->next != head && count<loc-1);

    if(count==loc-1)
    {
        if(current==tail)
        {
            delete_tail();
            return;
        }

        temp->next = current->next;
        current->next->prev = temp;
        free(current);
        return;
    }

    printf("Position (%d) does not exist!\n", loc);
}

void print_list()
{
    printf("FORWARD order print:\n");
```

Riyanshi

Riyanshi

```
    print_forward_order();

    printf("REVERSE order print:\n");
    print_reverse_order();
}

void print_forward_order()
{
    if(head==NULL)  return;

    struct node *current = head;

    do
    {
        printf("%d->", current->data);
        current = current->next;
    }   while(current != head);

    printf("\nList Length: %d\n", getListLength());
}


void print_reverse_order()
{
    if(head==NULL)  return;

    struct node *current = tail;

    do
    {
        printf("%d<-", current->data);
        current = current->prev;
    }   while(current != tail);

    printf("\nList Length: %d\n", getListLength());
    puts("\n");
}

//Determine the number of nodes in circular doubly linked list
int getListLength()
{
    if(head==NULL) return 0;

    int count = 0;
    struct node *current = head;

    do
    {
        count++;
        current = current->next;
    }   while(current != head);
```

Riyanshi

Riyanshi

```
   return count;
}

void main()
{
        head=NULL;int t,l,n,c;int a,b;
        int choice;
        do
        {
                printf("\nHERE IS THE CHOICE\n1:INSERT A NODE \n2:COUNT THE NUMBER OF 
NODES\n3.DELETE A NODE \n4:DISPLAY THE NODE\n5:QUIT");
                printf("\nEnter the choice:\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:printf("\n1.INSERT AT BEGINNING\n2.INSERT AT LAST\n3.INSERT AT 
PARTICULAR LOCATION");
                            printf("\nEnter your choice:\n");
                            scanf("%d",&t);
                            switch(t)
                            {
                                 case 1:insert_at_head();break;
                                         case 2:insert_at_tail();break;
                                         case 3:printf("\nEnter the location\n");
                                             scanf("%d",&l);
                                             insert_at_middle(l);
                                             break;
                                  }
                                  break;

                        case 2: c=getListLength();
                             printf("\nNumber of nodes are: %d",c);
                             break;
                        case 3:printf("\n1.DELETE AT BEGINNING\n2.DELETE AT LAST\n3.DELETE AT 
PARTICULAR LOCATION");
                            printf("\nEnter your choice:\n");
                            scanf("%d",&t);
                            switch(t)
                            {
                                 case 1:delete_head();break;
                                         case 2:delete_tail();break;
                                         case 3:printf("\nEnter the location\n");
                                             scanf("%d",&l);
                                             delete_middle(l);
                                             break;
                                  }
                                  break;
                        case 4: printf("LIST IS\n");
                             print_list();
                             break;

                  case 5: exit(0);
```

Riyanshi

Riyanshi

```
                        default:
                                printf("**********************\nWRONG
CHOICE\n**********************\n");

                }

        }while(choice!=5);
}
```

OUTPUT SAME AS LL

****

LINKED LIST HAVING HEAD AS LOCAL

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        int data;
        struct node *link;
};
void create_List(struct node **head,struct node **traverse);
void display(struct node *head);
int search(int no,struct node *head);
void exchange(struct node **head,struct node *change_to_head);

void swap(struct node **head,struct node *node1,struct node *node2);
int main(int argc, char const *argv[])
{
        int c=1,choice=1,val,pos;
        struct node *head=NULL,*traverse=NULL,*snode,*snode2;
        while(c)
        {
                printf("1:ENTER THE ELEMENT\n2:SEARCH\n3:DISPLAY\n4:QUIT\n5:MOVE TO
BEGINNING\n6:SWAP DATA\n");
                printf("ENTER YOUR CHOICE\n");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                create_List(&head,&traverse);
                        break;
                        case 2:
                                printf("ENTER NUMBER TO BE SEARCHED\n");
                                scanf("%d",&val);
                                pos=search(val,head);
                                if(pos==-1)
                                {
                                        printf("**********************\nELEMENT NOT
PRESENT\n**********************\n");
```

Riyanshi

Riyanshi

```
            continue;
        }
        printf("ELEMENT FOUND AT =%d\n", (pos+1));
break;
case 3:
        printf("LIST IS\n");
        display(head);
break;
case 4:
        c=0;
break;
case 5:
        printf("enter the node to be taken to begining\n");
        scanf("%d",&val);
        snode=head;
        while(snode!=NULL)
        {
            if(snode->data==val)
            {
                exchange(&head,snode);
                break;
            }
            snode=snode->link;
        }
        if(snode==NULL)
            printf("not FOUND\n");
break;
case 6:
        printf("ENTER VALUES TO BE EXCHANGED\n1:");
        scanf("%d",&val);
        snode=head;
        while(snode!=NULL)
        {
            if(snode->data==val)
            {
                break;
            }
            snode=snode->link;
        }
        if(snode==NULL)
            {
                printf("not FOUND\n");
                break;
            }
        printf("\n2:");
        scanf("%d",&val);
        snode2=head;
        while(snode2!=NULL)
        {
            if(snode2->data==val)
            {
                break;
```

Riyanshi

Riyanshi

```
                                        }
                                        snode2=snode2->link;
                                }
                                if(snode2==NULL)
                                        {
                                                printf("not FOUND\n");
                                                break;
                                        }
                                swap(&head,snode,snode2);
                                break;
                        default:
                                printf("WRONG CHOICE\n");
                }
        }
        return 0;
}
void create_List(struct node **head,struct node **traverse)
{
        struct node *nn;
        int val;


                nn=(struct node*)malloc(sizeof(struct node));
                printf("ENTER THE ELEMENT\n");
                scanf("%d",&val);
                nn->data=val;
                nn->link=NULL;
                if(*head==NULL)
                {
                *head=nn;
                *traverse=nn;
                }
                else
                {
                        (*traverse)->link=nn;
                        (*traverse)=(*traverse)->link;
                }


}
int search(int no,struct node *head)
{
        struct node *traverse;
        int pos=-1;
        traverse=head;
        while(traverse!=NULL)
        {
                pos++;
                if(traverse->data==no)
                        return pos;
                traverse=traverse->link;
```

Riyanshi

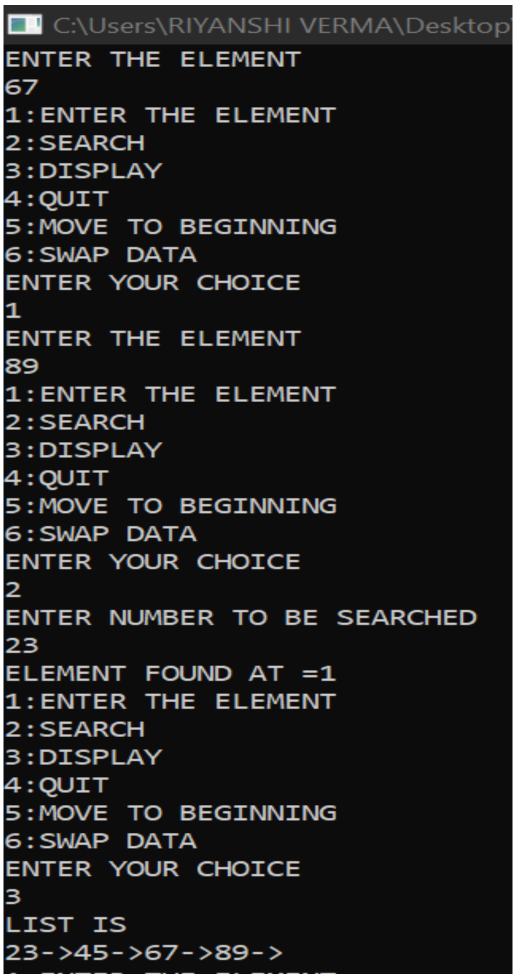
Riyanshi

```
        }
        return -1;

}
void display(struct node *head)
{
        struct node *traverse=head;
        if(head==NULL)
                printf("EMPTY LIST\n");
        while(traverse!=NULL)
        {
                        printf("%d->",traverse->data );
                        traverse=traverse->link;
        }
        printf("\n");
}
void exchange(struct node **head,struct node *change_to_head)
{
        struct node *ptr;
        ptr=*head;
        if(ptr==NULL)
        {
                *head=change_to_head;
        }
        while(ptr->link!=NULL)
        {

        printf("here\n");
                if(ptr->link==change_to_head)
                {
                        ptr->link=change_to_head->link;
                        change_to_head->link=*head;
                        *head=change_to_head;
                        break;
                }
                ptr=ptr->link;
        }

        display(*head);
}
void swap(struct node **head,struct node *node1,struct node *node2)
{
        struct node *ptr,*ptr1;
        ptr1=*head;
        ptr=*head;
        if(node1==*head)
        {
                ptr1=node1->link;
                while(ptr->link!=node2)
                        ptr=ptr->link;
                ptr->link=node1;
                node1->link=node2->link;
```

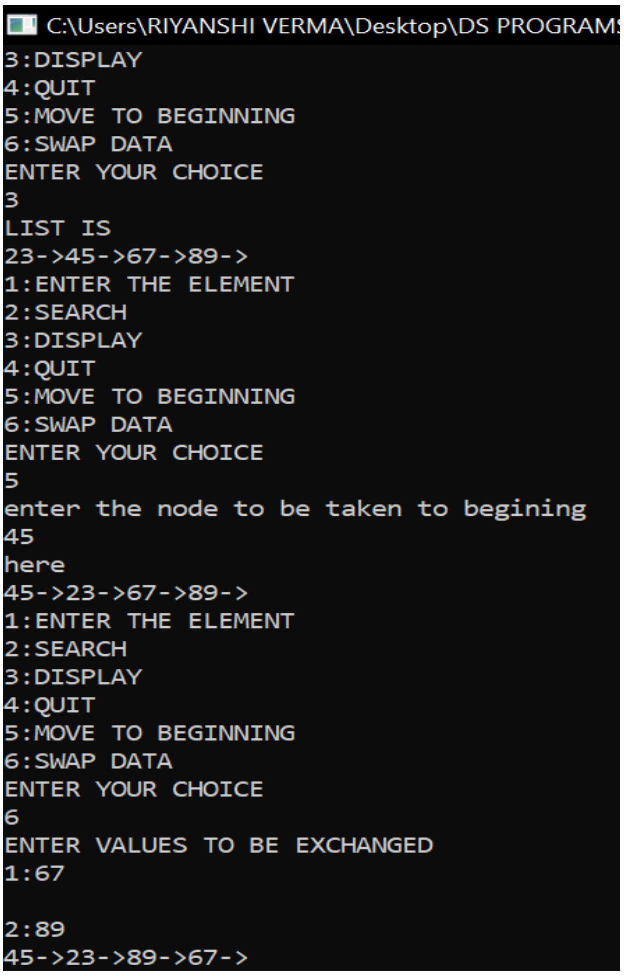
Riyanshi

Riyanshi

```
                node2->link=ptr1;
                *head=node2;
        }
        else
        {
                        while(ptr1->link!=node1)
                                ptr1=ptr1->link;
                        ptr=node1->link;
                        node1->link=node2->link;
                        node2->link=ptr;
                        while(ptr->link!=node2)
                        {
                                ptr=ptr->link;
                        }
                        ptr->link=node1;
                        ptr1->link=node2;
        }

        display(*head);
}
```



```
C:\Users\RIYANSHI VERMA\Desktop
ENTER THE ELEMENT
67
1:ENTER THE ELEMENT
2:SEARCH
3:DISPLAY
4:QUIT
5:MOVE TO BEGINNING
6:SWAP DATA
ENTER YOUR CHOICE
1
ENTER THE ELEMENT
89
1:ENTER THE ELEMENT
2:SEARCH
3:DISPLAY
4:QUIT
5:MOVE TO BEGINNING
6:SWAP DATA
ENTER YOUR CHOICE
2
ENTER NUMBER TO BE SEARCHED
23
ELEMENT FOUND AT =1
1:ENTER THE ELEMENT
2:SEARCH
3:DISPLAY
4:QUIT
5:MOVE TO BEGINNING
6:SWAP DATA
ENTER YOUR CHOICE
3
LIST IS
23->45->67->89->
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAM
3:DISPLAY
4:QUIT
5:MOVE TO BEGINNING
6:SWAP DATA
ENTER YOUR CHOICE
3
LIST IS
23->45->67->89->
1:ENTER THE ELEMENT
2:SEARCH
3:DISPLAY
4:QUIT
5:MOVE TO BEGINNING
6:SWAP DATA
ENTER YOUR CHOICE
5
enter the node to be taken to begining
45
here
45->23->67->89->
1:ENTER THE ELEMENT
2:SEARCH
3:DISPLAY
4:QUIT
5:MOVE TO BEGINNING
6:SWAP DATA
ENTER YOUR CHOICE
6
ENTER VALUES TO BE EXCHANGED
1:67

2:89
45->23->89->67->
```

****

<u>POLYNOMIAL ARITHMATICS</u>

Riyanshi

Riyanshi

1913075

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
	int coef,exp;
	struct node* next;
};
struct node* insert(struct node* head,int c,int e)
{
	struct node* prev,* cur;
	struct node* newnode=(struct node*)malloc(sizeof(struct node));
	newnode->coef=c;
	newnode->exp=e;
	if(head==NULL)
	{
		newnode->next=NULL;
		return newnode;
	}
	prev=cur=head;
	while(cur!=NULL && cur->exp >e)
	{
		prev=cur;
		cur=cur->next;
	}
	if(cur==head)
	{
		newnode->next=cur;
		return newnode;
	}
	else if(cur==NULL)
	{
		prev->next=newnode;
		newnode->next=NULL;
	}
	else
	{
		newnode->next=cur;
		prev->next=newnode;
	}
	return head;
}
struct node* append(struct node* head,int c,int e)
{
	struct node* newnode=(struct node*)malloc(sizeof(struct node));
	newnode->coef=c;
	newnode->exp=e;
	if(head==NULL)
	{
		newnode->next=NULL;
		return newnode;
	}
```

Riyanshi

Riyanshi

```
        struct node* trav=head;
        while(trav->next!=NULL)
        trav=trav->next;
        trav->next=newnode;
        newnode->next=NULL;
        return head;
}
struct node* polyadd(struct node* p1head, struct node* p2head)
{
        struct node* p3head=NULL;
        struct node* p1,* p2;
        int sumcoef;
        p1=p1head;
        p2=p2head;
        while(p1!=NULL && p2!=NULL)
        {
                if(p1->exp > p2->exp)
                {
                        p3head=append(p3head,p1->coef,p1->exp);
                        p1=p1->next;
                }
                else if(p1->exp < p2->exp)
                {
                        p3head=append(p3head,p2->coef,p2->exp);
                        p2=p2->next;
                }
                else
                {
                        sumcoef=(p1->coef)+(p2->coef);
                        if(sumcoef!=0)
                        {
                                p3head=append(p3head,sumcoef,p1->exp);
                        }
                        p1=p1->next;
                        p2=p2->next;
                }
        }
        while(p1!=NULL)
        {
                p3head=append(p3head,p1->coef,p1->exp);
                p1=p1->next;
        }
        while(p2!=NULL)
        {
                p3head=append(p3head,p2->coef,p2->exp);
                p2=p2->next;
  }
  return p3head;

}

void display(struct node* head)
```

Riyanshi

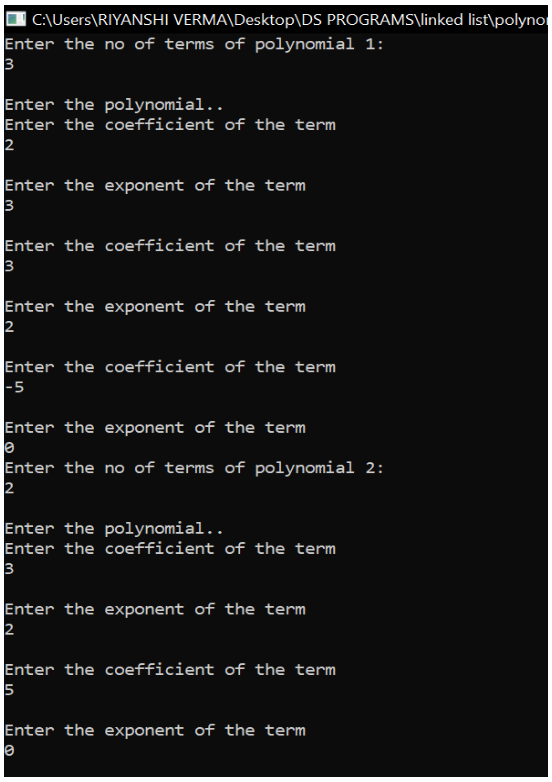
Riyanshi

```
{
            struct node *cur;
                        cur=head;
                        while(cur!=NULL)
                        {
                                    printf("%dx^%d",cur->coef,cur->exp);
                           if(cur->next != NULL)
          printf(" + ");
          cur=cur->next;
                        }
}



void main()
{
            int a,b,n,i;
            struct node* head1,* head2,* head3;
            head1=NULL;
            head2=NULL;
            printf("Enter the no of terms of polynomial 1:\n");
            scanf("%d",&n);
            printf("\nEnter the polynomial..");
            for(i=0;i<n;i++)
            {
                        printf("\nEnter the coefficient of the term\n");
                        scanf("%d",&a);
                        printf("\nEnter the exponent of the term\n");
                        scanf("%d",&b);
                        head1=insert(head1,a,b);
            }
            printf("Enter the no of terms of polynomial 2:\n");
            scanf("%d",&n);
            printf("\nEnter the polynomial..");
            for(i=0;i<n;i++)
            {
                        printf("\nEnter the coefficient of the term\n");
                        scanf("%d",&a);
                        printf("\nEnter the exponent of the term\n");
                        scanf("%d",&b);
                        head2=insert(head2,a,b);
            }
            head3=polyadd(head1,head2);

            printf("\nThe polynomial 1 is:\n");
            display(head1);
            printf("\nThe polynomial 2 is:\n");
            display(head2);
            printf("\n---------------------------------------------------------\n");
            printf("\nThe polynomial sum is:\n");
            display(head3);
}
```

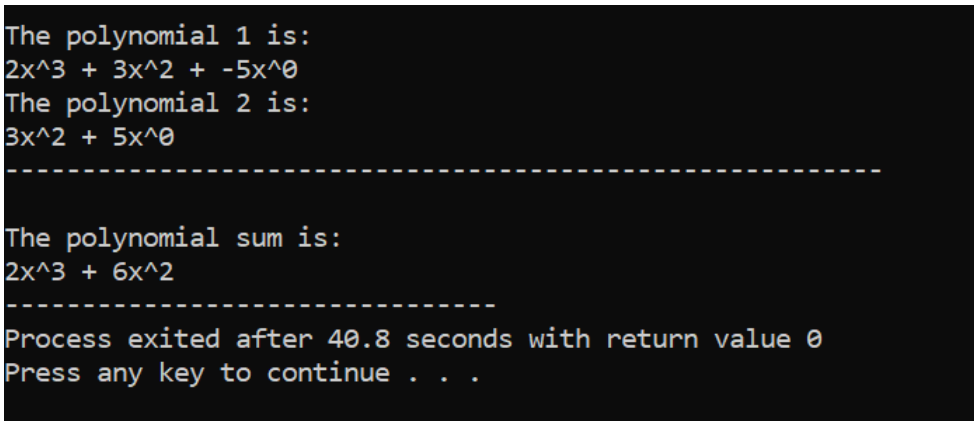
Riyanshi

Riyanshi

1913075



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\linked list\polyno
Enter the no of terms of polynomial 1:
3

Enter the polynomial..
Enter the coefficient of the term
2

Enter the exponent of the term
3

Enter the coefficient of the term
3

Enter the exponent of the term
2

Enter the coefficient of the term
-5

Enter the exponent of the term
0
Enter the no of terms of polynomial 2:
2

Enter the polynomial..
Enter the coefficient of the term
3

Enter the exponent of the term
2

Enter the coefficient of the term
5

Enter the exponent of the term
0
```



```
The polynomial 1 is:
2x^3 + 3x^2 + -5x^0
The polynomial 2 is:
3x^2 + 5x^0
----------------------------------------------------------

The polynomial sum is:
2x^3 + 6x^2
--------------------------------
Process exited after 40.8 seconds with return value 0
Press any key to continue . . .
```

****

BINARY SEARCH TREE

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
struct btree
{
	int n;
	struct btree *left;
	struct btree *right;
}typedef btree,node;
typedef struct snode
{
	node *ele;
	struct snode *next;
```

Riyanshi

Riyanshi

```
}snode;

snode *top;
node* topele();
int isempty();
node* pop();
void push(node *p);
node *create(node *);
node *insert(node *);
void search(node *);
node *modify(node *);
node *deletebycopy(node *);
void print(node *);
void inprint(node *);
void preprint(node *);
void postprint(node *);
void inordernr();
void preordernr();
void postordernr();

void main()
{
        int c;
        node *root;
        int num;
        while(c!=10)
        {
                printf("\n1. CREATE A BTREE");
                printf("\n2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER");
                printf("\n3. SEARCH A NODE");
                printf("\n4. INSERT A NEW NODE");
                printf("\n5. MODIFY ANY NODE");
                printf("\n6. DELETE ANY NODE");
                printf("\n7. PRINT INORDER NON-RECURSIVELY");
                printf("\n8. PRINT PREORDER NON-RECURSIVELY");
                printf("\n9. PRINT POSTORDER NON-RECURSIVELY");
                printf("\n10. EXIT");
                printf("\nENTER YOUR CHOICE:\n");
                scanf("%d",&c);
                switch(c)
                {
                        case 1: root=(node *)malloc(sizeof(node));
                                printf("\nEnter root:\n");
                                scanf("%d",&root->n);
                                root->left=root->right=NULL;
                                root=create(root);
                                break;
                        case 2: print(root);
                                break;
                        case 3: printf("\n");
                                search(root);
                                break;
```

Riyanshi

Riyanshi

```
                        case 4: printf("\n");
                                root =insert(root);
                                break;
                       case 5: printf("\n");
                               root =modify(root);
                               break;
                      case 6: printf("\n");
                               root =deletebycopy(root);
                               break;
                      case 7:
                               printf("\nNon recursive inorder traversal:\n");
                               inordernr(root);
                               break;
                      case 8:
                               printf("\nNon recursive preorder traversal:\n");
                               preordernr(root);
                               break;
                       case 9:
                               printf("\nNon recursive postorder traversal:\n");
                               postordernr(root);
                               break;
                     case 10:printf("\nTHANKYOU..");
                               exit(0);
                               break;
                      default:printf("\nINVALID CHOICE");

                   }

            }

}
node *topele()
{
            if(top!=NULL)
            return top->ele;
            else
            return NULL;
}
int isempty()
{
            return((top==NULL)?1:0);
}
void push(node *p)
{
            snode *temp;
            temp=(snode *)malloc(sizeof(snode));
            temp->ele=p;
            temp->next=top;
            top=temp;
}
node* pop()
{
```

Riyanshi

Riyanshi

1913075

```
   if(top!=NULL)
   {
           node *t;
           snode *temp;
           t=top->ele;
           temp=top;
           top=temp->next;
           return t;
           }
           else
           return NULL;
}
node *create(node *proot)
{
           char ch;
           int num;
           node *ptr, *prev;
           printf("Enter more nodes(y/n)?");
           ch=getche();
           while(ch=='y')
           {
                      ptr=prev=proot;
                      printf("\nEnter number : ");
                      scanf("%d",&num);
                      do
                      {
                                 if(num<ptr->n)
                                 {
                                            prev=ptr;
                                            ptr=ptr->left;
                                 }
                                 else if(num>ptr->n)
                                 {
                                            prev=ptr;
                                            ptr=ptr->right;
                                 }
                                 else
                                 {
                                            prev=NULL;
                                            break;
                                 }
                      }while(ptr);

                      if(prev)
                      {
                                 ptr=(node *)malloc(sizeof(node));
                                 ptr->n=num;
                                 ptr->left=ptr->right=NULL;
                                 if(ptr->n < prev->n)
                                 {
                                            prev->left=ptr;
                                 }
```

Riyanshi

Riyanshi

```
                    if(ptr->n > prev->n)
                    {
                              prev->right=ptr;
                              ptr->left=ptr->right=NULL;
                    }
          }
          else
             printf("%d is already present...",num);
             printf("\nEnter more nodes (y/n)?");
             ch=getche();
     }
     return proot;
}

node *insert(node *proot)
{
     int num;
     node *ptr,*prev;
     ptr=prev=proot;
     if(ptr==NULL)
     {
          printf("\nTree is empty..First create and then insert");
          return NULL;
     }
     printf("\nEnter number to be inserted : ");
     scanf("%d",&num);
     do{
          if(num < ptr->n)
          {
                    prev=ptr;
                    ptr=ptr->left;
          }
          else if(num > ptr->n)
          {
                    prev=ptr;
                    ptr=ptr->right;
          }
          else
          {
                    prev=NULL;
                    break;
          }
     }while(ptr!=NULL);
     if(prev!=NULL)
     {
          ptr=(node *)malloc(sizeof(node));
          ptr->n=num;
          ptr->left=ptr->right=NULL;
          if(ptr->n < prev->n)
          {
                    prev->left=ptr;
                    if(prev==proot)
```

Riyanshi

Riyanshi

```
                    proot=prev;
                }
                if(ptr->n > prev->n)
                {
                    prev->right=ptr;
                    ptr->left=ptr->right=NULL;
                    if(prev==proot)
                    proot=prev;
                }
                printf("%d is inserted..",num);
            }
            else
            printf("\n%d is already present..",num);
            return proot;
}

node *modify(node *proot)
{
            int mod,num;
            node *ptr,*succ,*prev,*p,*befr,*newnode;
            prev=ptr=proot;
            if(proot==NULL)
            {
                printf("\nTree is empty.... modification cannot be done in it");
                return NULL;
            }
            printf("\nEnter the number to get modified : ");
            scanf("%d",&num);
            printf("\nThen enter new number : ");
            scanf("%d",&mod);
            while(ptr!=NULL)
            {
                if(ptr->n==num)
                {
                    printf("\n%d present...modification can be done",num);
                    ptr=NULL;
                }
                else if(ptr->n<num)
                {
                    befr=ptr;
                    ptr=ptr->right;
                }
                else
                {
                    befr=ptr;
                    ptr=ptr->left;
              }
            }
                ptr=proot;
                while(ptr!=NULL)
                {
                    if(ptr->n == mod)
```

Riyanshi

Riyanshi

1913075

```
                        {
                                printf("\n %d already present... modification cannot be done!",mod);
                                return proot;
                        }
                        else if(ptr->n< mod)
                  {
                           ptr=ptr->right;
                  }
                  else
                  {
                           ptr=ptr->left;
          }
              }
              printf("\n %d not present... modification can be done!",mod);
              ptr=proot;
              prev=ptr;
     while(ptr)
     {
              if(ptr->n==num)
              {
                        if(!(ptr->left) && (ptr->right))
                        {
                                if(prev->left==ptr)
                                prev->left=NULL;
                                else
                                prev->right=NULL;


                                free(ptr);
                                return proot;
                        }
                        else if(!(ptr->left))
                        {
                                if(prev->left==ptr)
                                {
                                        prev->left=ptr->right;
                                   free(ptr);break;
                                }
                          else if(!(prev->right==ptr))
                          {
                                prev->right=ptr->right;
                                free(ptr);
                          }

                          return proot;
                  }
                  else if(!(ptr->right))
                  {
                        if(prev->left==ptr)
                                {
                                        prev->left=ptr->left;
                                   free(ptr);
```

Riyanshi

Riyanshi

```
                        }
                  else if(!(prev->right==ptr))
                  {
                        prev->right=ptr->left;
                        free(ptr);
                  }

                  return proot;
            }
            else
            {
                  p=ptr;
                  succ=ptr->right;
                  while(succ->left !=NULL)
                  {
                        p=succ;
                        succ=succ->left;
                        }
                        ptr->n=succ->n;
                        if(p->right==succ)
                        p->right=succ->right;
                        if(p->left==succ)
                        p->left=succ->right;

                        free(succ);
                  }

            }
            else if(ptr->n<num)
            {
                  prev=ptr;
                  ptr=ptr->right;
            }
            else
            {
                  prev=ptr;
                  ptr=ptr->left;
            }

      }
            newnode=(node *)malloc(sizeof(node));
            newnode->n=mod;
            newnode->left=newnode->right=NULL;
            prev=ptr=proot;
            do
            {
                  if(mod < ptr->n)
                  {
                        prev=ptr;
                        ptr=ptr->left;
                  }
                  else if(mod > ptr->n)
```

Riyanshi

Riyanshi

```
                    {
                            prev=ptr;
                            ptr=ptr->right;
                    }
                    else
                    {
                            prev=NULL;
                            break;
                    }
            }while(ptr!=NULL);
            if(prev!=NULL)
            {
                    if(newnode->n < prev->n)
                    prev->left=newnode;
                    if(newnode->n > prev->n)
                    prev->right=newnode;
                    printf("\n%d is inserted....",mod);

            }


    printf("\n%d is modified by %d",num,newnode->n);
    return proot;
}
node *deletebycopy(node *proot)
{
    node *p,*succ;
    int num;
    node *ptr=proot, *prev,*x;
    if(proot==NULL)
    {
            printf("\nTree is empty....");
            return NULL;
    }
    printf("\nEnter the number to be deleted:\n");
    scanf("%d",&num);
    prev=ptr;
    while(ptr)
    {
            if(ptr->n==num)
            {
                    if(!(ptr->left) && (ptr->right))
                    {
                            if(prev->left==ptr)
                            prev->left=NULL;
                            else
                            prev->right=NULL;

                            printf("\n%d is deleted....",num);
                            free(ptr);
                            return proot;
                    }
```

Riyanshi

Riyanshi

```
            else if(!(ptr->left))
            {
                    if(prev->left==ptr)
                    {
                            prev->left=ptr->right;
                      free(ptr);
                    }
              else if(!(prev->right==ptr))
              {
                    prev->right=ptr->right;
                    free(ptr);
              }
              printf("\n%d is deleted........",num);
              return proot;
      }
      else if(!(ptr->right))
      {
            if(prev->left==ptr)
                    {
                            prev->left=ptr->left;
                      free(ptr);
                    }
              else if(!(prev->right==ptr))
              {
                    prev->right=ptr->left;
                    free(ptr);
              }
              printf("\n%d is deleted........",num);
              return proot;
      }
      else
      {
            p=ptr;
            succ=ptr->right;
            while(succ->left !=NULL)
            {
                    p=succ;
                    succ=succ->left;
                    }
                    ptr->n=succ->n;
                    if(p->right==succ)
                    p->right=succ->right;
                    if(p->left==succ)
                    p->left=succ->right;
                    printf("\n%d is deleted........",num);
                    free(succ);
            }
            return proot;
    }
    else if(ptr->n<num)
    {
            prev=ptr;
```

Riyanshi

Riyanshi

```
                    ptr=ptr->right;
                }
                else
                {
                    prev=ptr;
                    ptr=ptr->left;
                }

        }
        printf("\nTree doesn't contain %d",num);
        return proot;
}
void search(node *proot)
{
        int num;
        node *ptr=proot;
        if(proot==NULL)
        {
                printf("\nTree is empty.. You cant search anything");
                return;
        }
        printf("\nEnter the number to be search : ");
        scanf("%d",&num);
        while(ptr)
        {
                if(ptr->n==num)
                {
                        printf("\n%d is found in BST", num);
                        return;
                }
                else if(ptr->n<num)
                ptr=ptr->right;
                else
                ptr=ptr->left;
        }
        printf("\nTree doesn't contain %d",num);
}
void inprint(node *ptr)
{
        if(!ptr)
        return;
        inprint(ptr->left);
        printf("%2d " ,ptr->n);
        inprint(ptr->right);
        return;
}
void preprint(node *ptr)
{
        if(!ptr)
        return;
        printf("%2d " ,ptr->n);
        preprint(ptr->left);
```

Riyanshi

Riyanshi

```
        preprint(ptr->right);
        return;
}
void postprint(node *ptr)
{
        if(!ptr)
        return;
        postprint(ptr->left);
        postprint(ptr->right);
        printf("%2d " ,ptr->n);
        return;
}

void print(node *proot)
{
        node *ptr=proot;
        if(!ptr)
        {
                printf("\nTree is empty");
                return;
        }
        printf("\nRoot is %d",proot->n);
        printf("\nINORDER : ");
        inprint(proot);
        printf("\nPREORDER : ");
        preprint(proot);
        printf("\nPOSTORDER : ");
        postprint(proot);
}
void inordernr(node *proot)
{
        node *p=proot;
        while(1)
        {
                if(p!=NULL)
                {
                        push(p);
                        p=p->left;
                }
                else if(isempty())
                {
                        return;
                }
                else
                {
                        p=pop();
                        printf("%d ",p->n);
                        p=p->right;
                }
        }
}
void preordernr(node *proot)
```

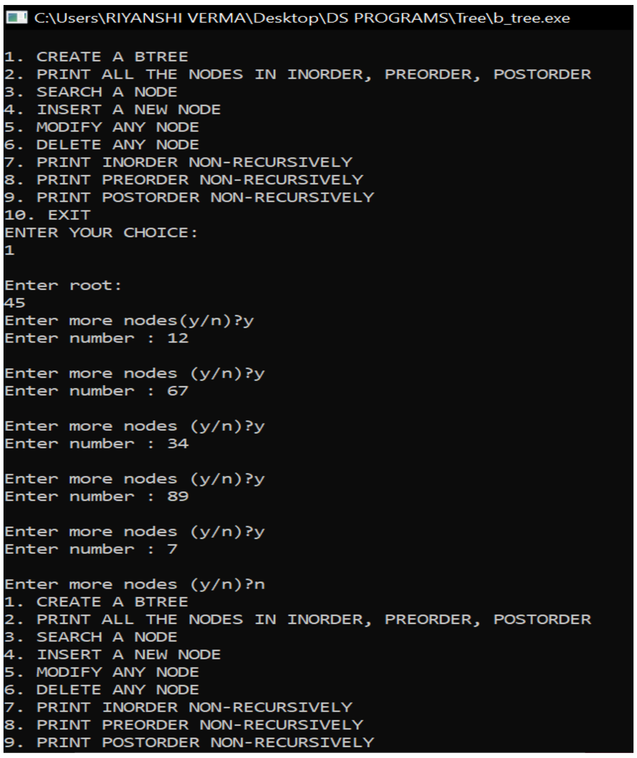
Riyanshi

Riyanshi

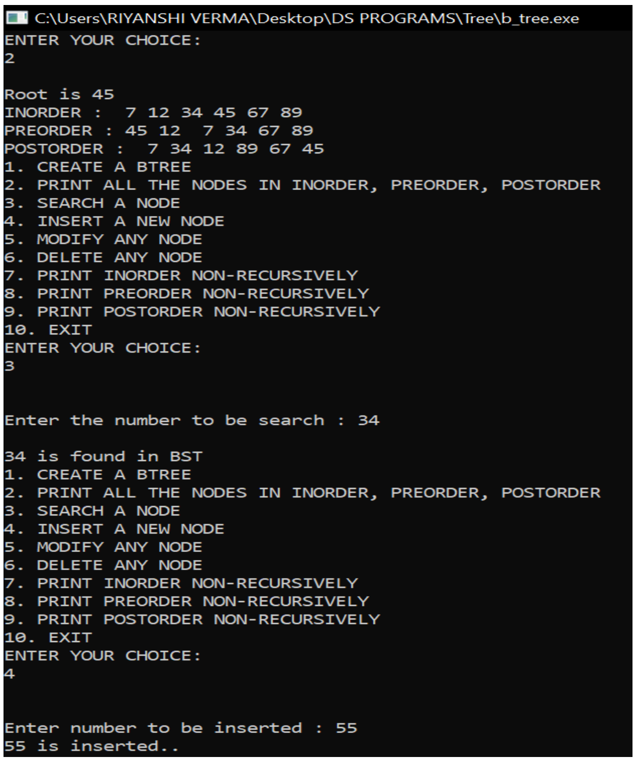
```
{
        node *t,*p=proot;
        top=NULL;
        if(proot==NULL)
        {
                printf("\nTree doesnt contain any node");
                return;
        }
        push(proot);
        while(isempty()==0)
        {
                t=pop();
                printf("%d ",t->n);
                if(t->right)
                  push(t->right);
                if(t->left)
                  push(t->left);
        }
}

void postordernr(node *proot)
{
        node *p=proot;
        if(p==NULL)
        return;
        top=NULL;
        do
        {
                while(p)
                {
                        if(p->right)
                          push(p->right);
                        push(p);
                        p=p->left;
                }
                        p=pop();
                        if(p->right && topele()==p->right)
                        {
                                pop();
                                push(p);
                                p=p->right;
                        }
                        else
                        {
                                printf("%d ",p->n);
                                p=NULL;
                        }
                }while(!isempty());
        }
```

Riyanshi

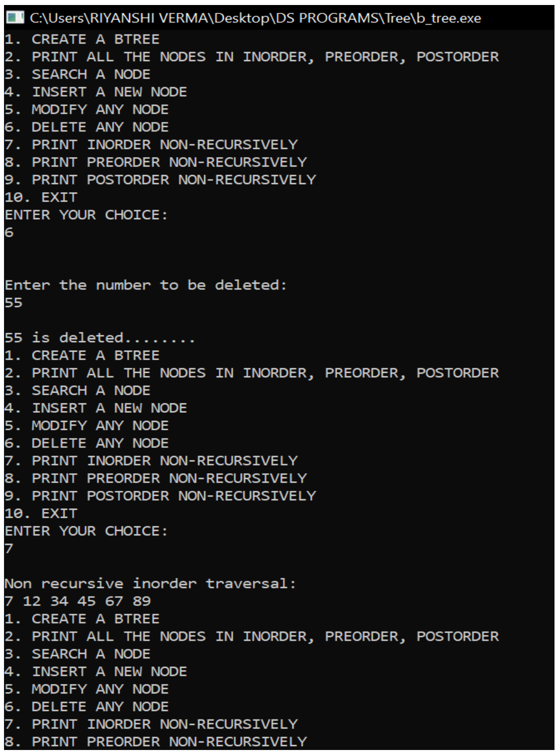
Riyanshi



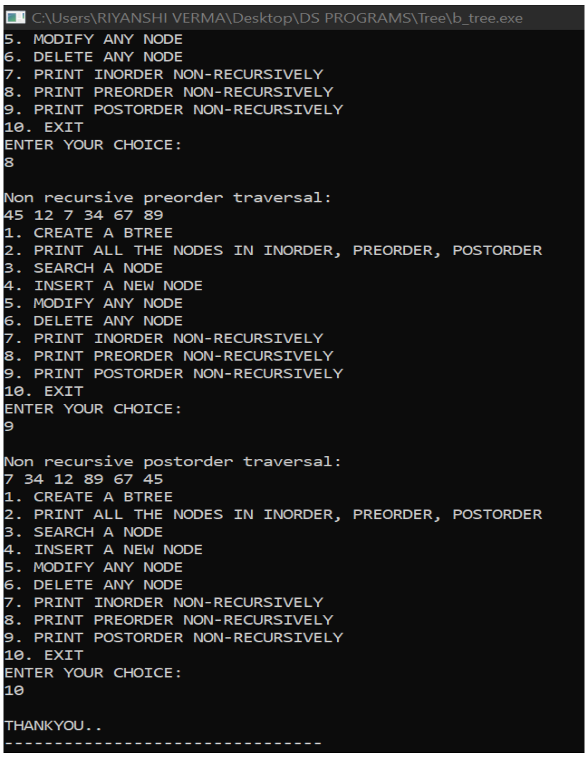
```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\Tree\b_tree.exe

1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
1

Enter root:
45
Enter more nodes(y/n)?y
Enter number : 12

Enter more nodes (y/n)?y
Enter number : 67

Enter more nodes (y/n)?y
Enter number : 34

Enter more nodes (y/n)?y
Enter number : 89

Enter more nodes (y/n)?y
Enter number : 7

Enter more nodes (y/n)?n
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\Tree\b_tree.exe
ENTER YOUR CHOICE:
2

Root is 45
INORDER :  7 12 34 45 67 89
PREORDER : 45 12  7 34 67 89
POSTORDER :  7 34 12 89 67 45
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
3


Enter the number to be search : 34

34 is found in BST
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
4


Enter number to be inserted : 55
55 is inserted..
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\Tree\b_tree.exe
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
6


Enter the number to be deleted:
55

55 is deleted........
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
7

Non recursive inorder traversal:
7 12 34 45 67 89
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\Tree\b_tree.exe
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
8

Non recursive preorder traversal:
45 12 7 34 67 89
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
9

Non recursive postorder traversal:
7 34 12 89 67 45
1. CREATE A BTREE
2. PRINT ALL THE NODES IN INORDER, PREORDER, POSTORDER
3. SEARCH A NODE
4. INSERT A NEW NODE
5. MODIFY ANY NODE
6. DELETE ANY NODE
7. PRINT INORDER NON-RECURSIVELY
8. PRINT PREORDER NON-RECURSIVELY
9. PRINT POSTORDER NON-RECURSIVELY
10. EXIT
ENTER YOUR CHOICE:
10

THANKYOU..
--------------------------------
```

****

HEAP SORT

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

Riyanshi

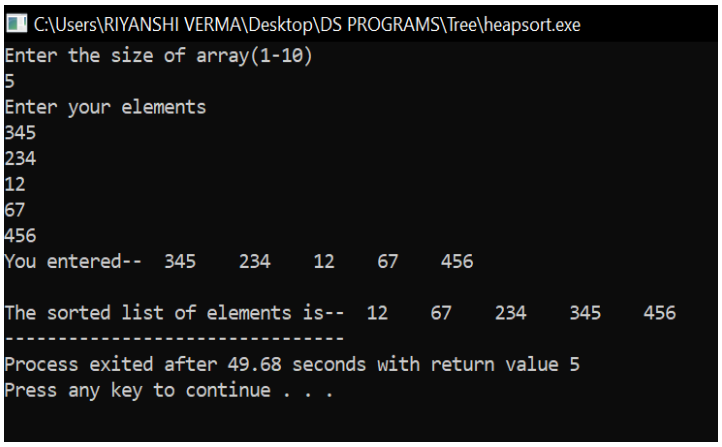
Riyanshi

```
void swap(int *i, int *j)
{
  int t=*i;
  *i=*j;
  *j=t;
}
void heapify(int arr[],int n,int i)
{
        int largest=i;
        int l = 2*i+1;
        int r = 2*i+2;
        if(l<n && arr[l] > arr[largest])
        largest=l;
        if(r<n && arr[r] > arr[largest])
        largest=r;
        if(largest !=i)
        {
                swap(&arr[i],&arr[largest]);
                heapify(arr,n,largest);
        }
}

void heapSort(int arr[],int n)
{
        int i;
        for( i=n/2-1;i>=0;i--)
        heapify(arr,n,i);
        for(i=n-1;i>0;i--)
        {
                swap(&arr[0],&arr[i]);
                heapify(arr,i,0);
        }
}



void main()
{
        int n;
        printf("Enter the size of array(1-10)\n");
        scanf("%d",&n);
        if(n<0 || n>10)
        {
                printf("Enter correct value of n\n ");
                scanf("%d",&n);
        }
        if(n<0 || n>10)
        {
                exit(0);
        }
        int arr[10];
```

Riyanshi

Riyanshi

```
        int i;
        printf("Enter your elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&arr[i]);
        }
        printf("You entered--  ");
        for(i=0;i<n;i++)
        {
                printf("%d    ",arr[i]);
        }
        heapSort(arr,n);
        printf("\n\nThe sorted list of elements is--  ");
        for(i=0;i<n;i++)
        {
                printf("%d    ",arr[i]);
        }
}
```



```
C:\Users\RIYANSHI VERMA\Desktop\DS PROGRAMS\Tree\heapsort.exe
Enter the size of array(1-10)
5
Enter your elements
345
234
12
67
456
You entered--  345    234    12    67    456

The sorted list of elements is--  12    67    234    345    456
--------------------------------
Process exited after 49.68 seconds with return value 5
Press any key to continue . . .
```

****

RADIX SORT

```
#include<stdio.h>
int get_max (int a[], int n)
{
      int max = a[0];
      int i;
      for ( i=1; i<n; i++)
           if (a[i] > max)
           max=a[i];
     return max;
}
void radix_sort (int a[], int n)
{
      int bucket[10][10], bucket_cnt[10];
      int i,j,k,r,x,countdigit=0,divisor=1,lar,pass;
      lar = get_max (a, n);
```

Riyanshi

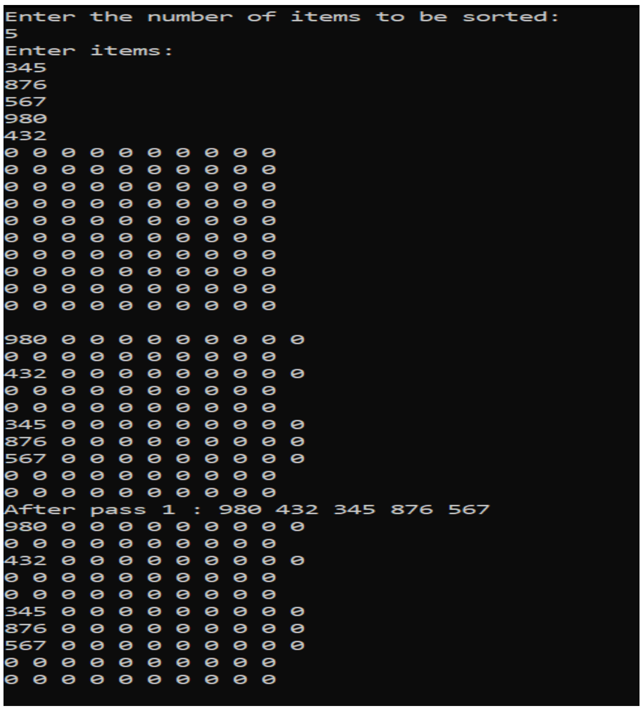
Riyanshi

```
  while (lar>0)
  {
        countdigit++;
        lar /= 10;
  }
 for (k = 0; k < 10; k++)
     {
        for (j = 0; j < 10; j++)
            {
               bucket[k][j]=0;
            }

     }

  for (pass=0;pass<countdigit;pass++)
      {
         for (i=0;i<10;i++)
              {
                 bucket_cnt[i] = 0;
              }

   for (k = 0; k < 10; k++)
       {
           for (j = 0; j < 10; j++)
              {
                  printf("%d ",bucket[k][j]);
              }
           printf("\n");
       }
printf("\n");
for (i=0; i<n; i++)
     {
           r = (a[i] / divisor) % 10;
           x=bucket_cnt[r];
           bucket[r][x] = a[i];
           bucket_cnt[r] += 1;
     }
i = 0;
for (k = 0; k < 10; k++)
     {
          for (j = 0; j < bucket_cnt[k]; j++)
              {
                   a[i] = bucket[k][j];
                   i++;
              }
      }
 for (k = 0; k < 10; k++)
     {
         for (j = 0; j < 10; j++)
             {
                printf("%d ",bucket[k][j]);
             }
```

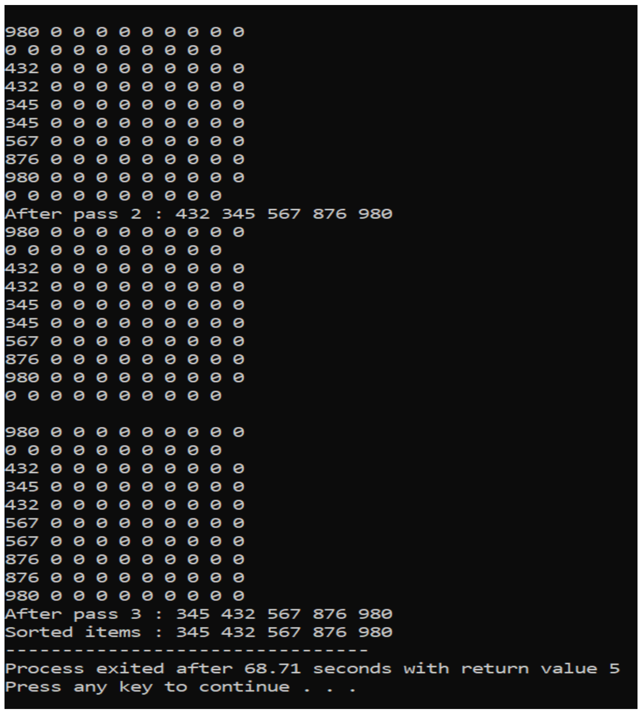
Riyanshi

Riyanshi

```
                printf("\n");
            }
      divisor *= 10;
      printf ("After pass %d : ", pass + 1);
      for (i = 0; i < n; i++)
            printf ("%d ", a[i]);
      printf ("\n");
   }
}
void main ()
{
            int i, n, a[10];
            printf ("Enter the number of items to be sorted: \n");
            scanf ("%d", &n);
            printf ("Enter items: \n");
            for (i = 0; i < n; i++)
              {
                  scanf ("%d", &a[i]);
              }
             radix_sort (a, n);
             printf ("Sorted items : ");
             for (i = 0; i < n; i++)
                  printf ("%d ", a[i]);
}
```



```
Enter the number of items to be sorted:
5
Enter items:
345
876
567
980
432
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

980 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
345 0 0 0 0 0 0 0 0 0
876 0 0 0 0 0 0 0 0 0
567 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
After pass 1 : 980 432 345 876 567
980 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
345 0 0 0 0 0 0 0 0 0
876 0 0 0 0 0 0 0 0 0
567 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
```



```
980 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
345 0 0 0 0 0 0 0 0 0
345 0 0 0 0 0 0 0 0 0
567 0 0 0 0 0 0 0 0 0
876 0 0 0 0 0 0 0 0 0
980 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
After pass 2 : 432 345 567 876 980
980 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
345 0 0 0 0 0 0 0 0 0
345 0 0 0 0 0 0 0 0 0
567 0 0 0 0 0 0 0 0 0
876 0 0 0 0 0 0 0 0 0
980 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

980 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
345 0 0 0 0 0 0 0 0 0
432 0 0 0 0 0 0 0 0 0
567 0 0 0 0 0 0 0 0 0
567 0 0 0 0 0 0 0 0 0
876 0 0 0 0 0 0 0 0 0
876 0 0 0 0 0 0 0 0 0
980 0 0 0 0 0 0 0 0 0
After pass 3 : 345 432 567 876 980
Sorted items : 345 432 567 876 980
--------------------------------
Process exited after 68.71 seconds with return value 5
Press any key to continue . . .
```

************************************END**

Riyanshi