

Lab Program 1 :-

WAP to simulate the working of Stack using an array with the following A:- Push , B:- Pop & display. The program should front appropriate

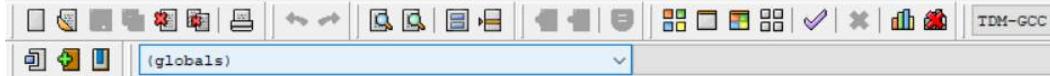
```
#include <stdio.h>
#include <process.h>
#include <conio.h>
#define STACK_SIZE 5
int top = -1;
int s[10];
int item;
void push()
{
    if (top == STACK_SIZE - 1)
    {
        printf(" stack overflow \n");
        return;
    }
    top = top + 1;
    s[top] = item;
}
int pop()
{
    if (top == -1) return -1;
    return s[top--];
}
void display()
{
    int i;
    if (top == -1)
    {
        printf(" stack is empty \n");
        return;
    }
    printf(" contents of the stack \n");
}
```

```

for (i = top; i >= 0; i--)
{
    printf ("%d\n", s[i]);
}
int main()
{
    int item_deleted;
    char choice;
    clrscr();
    for (;;)
    {
        printf ("In A : push In B : pop \n : 0 : display \n 1 : exit\n");
        printf ("Enter the choice \n");
        scanf ("%c", &choice);
        switch (choice)
        {
            case 'A':
                printf ("Enter the item to be inserted \n");
                scanf ("%d", &item);
                push();
                break;
            case 'B':
                item_deleted = pop();
                if (item_deleted == -1)
                    printf ("Stack is empty\n");
                else
                    printf ("item deleted is %d\n", item_deleted);
                break;
            case 'C':
                display();
                break;
            default:
                exit(0);
        }
    }
    getch();
}

```

File Edit Search View Project Execute Tools AStyle Window Help



TDM-GCC 4.9.2 64-bit Release

push and pop.cpp

```
1 #include<stdio.h>
2 #include<process.h>
3 #include<conio.h>
4 #define STACK_SIZE 5
5 int top=-1;
6 int s[10];
7 int item;
8 void push()
9 {
10    if(top==STACK_SIZE-1){printf("stack overflow\n");
11    return;}
12    top=top+1;
13    s[top]=item;
14 }
15 int pop()
16 { if(top==-1) return -1;
17 return s[top--];
18 }
19 void display()
20 { int i;
21 if(top==-1)
22 {printf(" Stack is empty \n");
23 return;
24 }
25 printf("contents of the stack\n");
26 for(i=top; i>=0; i--)
27 { printf("%d\n", s[i]);
28 }
29 }
30 int main()
31 {
32    int item_deleted;
33    int choice;
34    system("cls");
35    for(;;)
36    {
37        printf("\n 1: Push\n 2: Pop\n 3: Display\n 4: Exit\n");
38    }
```

Compiler Resources Compile Log Debug Find Results

Line: 11 Col: 12 Sel: 0 Lines: 61 Length: 1123 Insert Done parsing in 0.11 seconds



14:32 05-10-2020

The screenshot shows the Dev-C++ IDE interface with the following details:

- Title Bar:** C:\Users\sohan\Desktop\C Programs\Data Structures Lab\LP 1\push and pop.cpp - Dev-C++ 5.11
- Menu Bar:** File, Edit, Search, View, Project, Execute, Tools, AStyle, Window, Help
- Toolbar:** Includes icons for file operations like Open, Save, Print, and Build.
- Compiler Selection:** TDM-GCC 4.9.2 64-bit Release
- Code Editor:** The main window displays the C code for "push and pop.cpp". The code implements a stack using an array and handles four choices: Push, Pop, Display, and Exit.

```
25 }
26 printf("contents of the stack\n");
27 for(i=top; i>=0; i--)
28 { printf("%d\n", s[i]);
29 }
30 }
31 int main()
32 {
33     int item_deleted;
34     int choice;
35     system("cls");
36     for(;;)
37     {
38         printf("\n 1: Push\n 2: Pop\n 3: Display\n 4: Exit\n");
39         printf(" Enter the choice\n");
40         scanf("%d", &choice);
41         switch(choice)
42         {
43             case 1: printf("enter the item to be inserted \n");
44             scanf("%d\n", &item);
45             push();
46             break;
47             case 2: item_deleted=pop(); (-1);
48             if(item_deleted== -1)
49             {printf("Stack is empty\n");
50             }
51             else
52             printf("item deleted is %d\n", item_deleted);
53             break;
54             case 3: display();
55             break;
56             default: exit(0);
57         }
58     }
59     getch();
60     return 0;
61 }
```

- Toolbars:** Compiler, Resources, Compile Log, Debug, Find Results
- Status Bar:** Line: 11, Col: 12, Sel: 0, Lines: 61, Length: 1123, Insert, Done parsing in 0.11 seconds
- Taskbar:** Shows various application icons including File Explorer, Task View, Mail, Settings, and DEV.
- System Tray:** Displays icons for battery, signal strength, ENG, 14:32, 05-10-2020, and a notification bubble.

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\LP 1\push and pop.exe

```
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
1  
enter the item to be inserted  
12  
1  
  
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
enter the item to be inserted  
15  
1  
  
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
enter the item to be inserted  
43  
1  
  
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
enter the item to be inserted  
32  
1  
  
1:Push  
2:Pop  
3:Display
```



14:32
05-10-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\LP 1\push and pop.exe

```
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
enter the item to be inserted  
2
```

```
2  
stack overflow
```

```
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
item deleted is 32
```

```
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
3  
contents of the stack  
43  
15  
12  
12
```

```
1:Push  
2:Pop  
3:Display  
4:Exit  
Enter the choice  
4
```

```
-----  
Process exited after 170.3 seconds with return value 0  
Press any key to continue . . .
```



Type here to search



14:32
05-10-2020

Lab Program 2:

Q) Write a program to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and binary operators + (plus), - (minus), * (multiply) & / (divide).

```
A. #include <stdio.h>
#include <conio.h>
#include <string.h>
#include <process.h>
int FC( char symbol )
{ Switch ( symbol )
  { case '+':
    case '-': return 2;
    case '*':
    case '/': return 4;
    case '^':
    case '$': return 5;
    case 'C': return 0;
    case '#': return -1;
    default : return 8;
  }
}
```

```
int GC( char symbol )
{ Switch ( symbol )
  { case '+':
    case '-': return 1;
    case '*':
    case '/': return 3;
    case '^':
    case '$': return 6;
    case 'C': return 9;
    case ')': return 0;
    default : return 7;
  }
}
```

ff

P.T.O.

```
void infix_postfix(char infix[], char postfix[])
```

```
{ int top, i, j; 
```

```
char s[30], symbol; 
```

```
top = -1; 
```

```
s[++top] = '#'; 
```

```
j=0; 
```

```
for(i=0; i<strlen(infix); i++) 
```

```
{ symbol = infix[i]; 
```

```
while(F(s[top]) > G(symbol)) 
```

```
{ 
```

```
postfix[j] = s[top--]; 
```

```
j++; 
```

```
{ 
```

```
if(F(s[top]) != G(symbol)) 
```

```
s[++top] = symbol; 
```

```
else 
```

```
{ top--; 
```

```
{ 
```

```
while(s[top] != '#') 
```

```
{ 
```

```
postfix[j++] = s[top--]; 
```

```
{ 
```

```
postfix[j] = '\0'; 
```

```
{ 
```

```
int main() 
```

```
{ int t; 
```

```
char infix[20]; 
```

```
char postfix[20]; 
```

```
system("cls"); 
```

```
printf("Enter the valid infix expression.\n"); 
```

```
scanf("%s", infix); 
```

```
for(t=0; t<strlen(infix); t++) { 
```

```
if(infix[t] == '+' || infix[t] == '-' || infix[t] == '*' ||  
infix[t] == '/' || infix[t] == '^' || infix[t] == '(')
```

```
{  
if(infix[t+1] == '+' || infix[t+1] == '-' || infix[t+1] == '*' ||  
infix[t+1] == '/' || infix[t+1] == '^' || infix[t+1] == ')',
```

```
printf("Invalid"); exit(0);
```

```
???
```

```
infix_postfix(infix, postfix);  
printf("the postfix exp is \n");  
getch();  
return 0;
```

```
f
```



TDM-GCC 4.9.2 64-bit Release



(globals)

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<string.h>
4 #include<process.h>
5 int F(char symbol)
6 {
7     switch(symbol)
8     {
9         case '+': return 2;
10        case '-': return 2;
11        case '*': return 4;
12        case '/': return 4;
13        case '^': return 5;
14        case '$': return 5;
15        case '(': return 0;
16        case ')': return -1;
17        default: return 8;
18    }
19 }
20 int G(char symbol)
21 {
22     switch(symbol)
23     {
24         case '+': return 1;
25         case '-': return 1;
26         case '*': return 3;
27         case '/': return 3;
28         case '^': return 6;
29         case '$': return 6;
30         case '(': return 9;
31         case ')': return 0;
32         default: return 7;
33     }
34 }
35 void infix_postfix(char infix[], char postfix[])
36 {
37     int top,i,j;
38     char s[30],symbol;
```



Line: 58 Col: 2 Sel: 0 Lines: 83 Length: 1356 Insert Done parsing in 0.015 seconds



Type here to search

15:52
05-10-2020



TDM-GCC 4.9.2 64-bit Release

(globals)

```
LP_2.cpp
35 void infix_postfix(char infix[], char postfix[])
36 {
37     int top, i, j;
38     char s[30], symbol;
39     top = -1;
40     s[++top] = '#';
41     j = 0;
42     for(i=0; i < strlen(infix); i++)
43     {
44         symbol = infix[i];
45         while(F(s[top]) > G(symbol))
46         {
47             postfix[j] = s[top--];
48             j++;
49         }
50         if(F(s[top]) == G(symbol))
51             s[++top] = symbol;
52         else
53             top--;
54     }
55     while(s[top] != '#')
56     {
57         postfix[j++] = s[top--];
58     }
59     postfix[j] = '\0';
60 }
61 int main()
62 {
63     int t;
64     char infix[20];
65     char postfix[20];
66     system("cls");
67     printf("Enter the valid infix expression\n");
68     scanf("%s", infix);
69     for(t=0; t < strlen(infix); t++)
70     {
71         if(infix[t] == '+' || infix[t] == '-' || infix[t] == '*' || infix[t] == '/' || infix[t] == '^' || infix[t] == '(')
72         {
73             if(infix[t+1] == '+' || infix[t+1] == '-' || infix[t+1] == '*' || infix[t+1] == '/' || infix[t+1] == '^' || infix[t+1] == ')')
```

Compiler Resources Compile Log Debug Find Results

Line: 58 Col: 2 Sel: 0 Lines: 83 Length: 1356 Insert Done parsing in 0.015 seconds



Type here to search



15:53 ENG 05-10-2020

File Edit Search View Project Execute Tools AStyle Window Help



TDM-GCC 4.9.2 64-bit Release

```
LP_2.cpp
47 postfi x[ j ] = s[ top-- ];
48 j++;
49 }
50 if( F( s[ top ] ) != G( symbol ) )
51 s[ ++top ] = symbol ;
52 else
53 top--;
54 }
55 while( s[ top ] != '#' )
56 {
57 postfi x[ j ++] = s[ top-- ];
58 }
59 postfi x[ j ] = '\0' ;
60 }
61 int main()
62 {
63 char infix[ 20 ];
64 char postfix[ 20 ];
65 system( "cls" );
66 printf( "Enter the valid infix expression\n" );
67 scanf( "%s", infix );
68 for( t=0; t<strlen(infix); t++ )
69 {
70 if( infix[ t ] == '+' || infix[ t ] == '-' || infix[ t ] == '*' || infix[ t ] == '/' || infix[ t ] == '^' || infix[ t ] == '(' )
71 {
72 if( infix[ t+1 ] == '+' || infix[ t+1 ] == '-' || infix[ t+1 ] == '*' || infix[ t+1 ] == '/' || infix[ t+1 ] == '^' || infix[ t+1 ] == ')' )
73 {
74 printf( "Inval id" );
75 exit( 0 );
76 }
77 infix_postfix( infix, postfix );
78 printf( "the postfix exp is\n" );
79 printf( "%s \n", postfix );
80 getch();
81 return 0;
82 }
83 }
```

Compiler Resources Compile Log Debug Find Results

Line: 58 Col: 2 Sel: 0 Lines: 83 Length: 1356 Insert Done parsing in 0.015 seconds



Type here to search

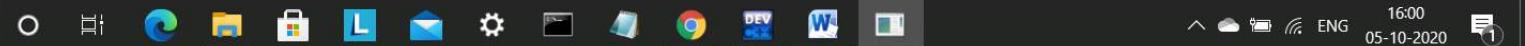


15:53 ENG 05-10-2020 1

```
C:\Users\sohan\Desktop\C Programs\Data Structures Lab\LP 2\dev\LP_2.exe
Enter the valid infix expression
a+b*c(/d)
Invalid
-----
Process exited after 36.19 seconds with return value 0
Press any key to continue . . .
```



Type here to search



16:00
ENG
05-10-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\LP 2\dev\LP_2.exe

Enter the valid infix expression

((a*b)^(c/d))-g

the postfix exp is

ab*cd/^g-



Type here to search



16:01
05-10-2020



Lab Program 3:-

38 Write a program to simulate the working of a queue of integers using an array. Provide the following operations.

a) Push, Top, Pop, Insert, Delete, Display.

The program should print appropriate messages for queue empty & queue overflow conditions.

A>

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

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

```
#define QUE_SIZE 3
```

```
int item, front = 0, rear = -1, q[10];
```

```
void insertrear()
```

```
if (rear == QUE_SIZE - 1)
```

```
printf("queue overflow\n");
```

```
return;
```

```
}
```

```
rear = rear + 1;
```

```
q[rear] = item;
```

```
}
```

```
int deletefront()
```

```
if (front > rear)
```

```
front = 0;
```

```
rear = -1;
```

```
return -1;
```

```
}
```

```
return q[front++];
```

```
}
```

```
void display()
```

```
int i;
```

```
if (front > rear)
```

```
printf("queue is empty\n");
```

```
return;
```

```
}
```

```
printf("contents of queue
```

```
is empty is empty \n");
```

```
for(i=fant; i<=rear; i++) {
```

```
    printf ("%d\n", q[i]);
```

```
}}
```

```
int main() {
```

```
    int choice;
```

```
    for(;;) {
```

```
        printf ("1: insert rear \n 2: delete front \n 3: display \n 4: exit \n"),
```

```
        printf ("enter your choice \n"),
```

```
        scanf ("%d", &choice),
```

```
        switch (choice) {
```

```
            case 1: printf ("enter item to be inserted \n"),
```

```
            scanf ("%d", &item),
```

```
            insert_rear(),
```

```
            break;
```

```
            case 2: item = delete_front();
```

```
            if(item == -1)
```

```
                printf ("queue is empty \n"),
```

```
            else
```

```
                printf ("item deleted = %d \n", item);
```

```
            break;
```

```
            case 3: displayQ();
```

```
            break;
```

```
            default: exit(0);
```

```
}}
```

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linear queue\linear_q.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

linear_q.cpp

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #define QUE_SIZE 3
4 int item, front=0, rear=-1, q[10];
5 void insertrear()
6 {
7     if(rear==QUE_SIZE-1)
8     {
9         printf("queue overflow\n");
10    return;
11 }
12 rear=rear+1;
13 q[rear]=item;
14 int deletefront()
15 {
16     if(front>rear)
17     {
18         front=0;
19         rear=-1;
20         return -1;
21     }
22     return q[front++];
23 }
24 void displayQ()
25 {
26     if(front>rear)
27     {
28         printf("queue is empty\n");
29     }
30     printf("contents of queue\n");
31     for(i=front;i<=rear;i++)
32     {
33         printf("%d\n", q[i]);
34     }
35     main()
36 {
37     int choice;
38     for(;;)
39     {
40         printf("1:insertrear 2:deletefront 3:display 4:exit\n");
41         printf("enter the choice\n");
42     }
43 }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Sel: 0 Lines: 59 Length: 982 Insert Done parsing in 0.109 seconds

Type here to search

Windows Start File Explorer Task View Settings File Explorer Mail Edge Task View Chrome Word DEV

12:03 ENG 06-11-2020 6

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linear queue\linear_q.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
linear_q.cpp

23 {
24     printf("queue is empty\n");
25     return;
26 }
27 printf("contents of queue\n");
28 for(i=front;i<=rear;i++)
29 {
30     printf("%d\n", q[i]);
31 }
32 int main()
33 {
34     int choice;
35     for(;;)
36     {
37         printf("1:insert rear 2:delete front 3:display 4:exit\n");
38         printf("enter the choice\n");
39         scanf("%d", &choice);
40         switch(choice)
41         {
42             case 1:printf("enter the item to be inserted\n");
43             scanf("%d", &item);
44             insertrear();
45             break;
46             case 2:item=deletefront();
47             if(item== -1)
48                 printf("queue is empty\n");
49             else
50                 printf("item deleted=%d\n", item);
51             break;
52             case 3:displayQ();
53             break;
54             default:exit(0);
55         }
56     }
57 }
58 }
59 }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Sel: 0 Lines: 59 Length: 982 Insert Done parsing in 0.109 seconds

Type here to search

12:03 ENG 06-11-2020 6

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linear queue\linear_q.exe

```
1:insertrear 2:deletefront 3:display 4:exit
enter the choice
1
enter the item to be inserted
12
1:insertrear 2:deletefront 3:display 4:exit
enter the choice
1
enter the item to be inserted
32
1:insertrear 2:deletefront 3:display 4:exit
enter the choice
1
enter the item to be inserted
55
1:insertrear 2:deletefront 3:display 4:exit
enter the choice
1
enter the item to be inserted
3
queue overflow
1:insertrear 2:deletefront 3:display 4:exit
enter the choice
2
item deleted=12
1:insertrear 2:deletefront 3:display 4:exit
enter the choice
3
contents of queue
32
55
1:insertrear 2:deletefront 3:display 4:exit
enter the choice
4
```

Process exited after 29.8 seconds with return value 0
Press any key to continue . . .



Lab Program 4:

4E Write a program to simulate the working of a circular queue of integers using an array. Provide the following operations.

a) Insert & Delete or Display.

The program should print appropriate messages for queue empty and queue overflow conditions.

A:

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

```
#define que_size 3
```

```
int item, front = 0, rear = -1, q[que_size], count = 0;
```

```
void insertrear()
```

```
if (count == que_size)
```

```
printf("queue overflow");
```

```
return;
```

}

```
rear = (rear + 1) % que_size;
```

```
q[rear] = item;
```

```
count++;
```

}

```
int deletefront()
```

```
if (count == 0) return -1;
```

```
item = q[front];
```

```
front = (front + 1) % que_size;
```

```
count = count - 1;
```

```
return item;
```

g

```
void displayq()
```

```
int i, f;
```

```
if (count == 0)
```

```
printf("queue is empty");
```

```
return;
```

g

```

f = front;
printf("contents of queue in ");
for(i=0; i<count; i++) {
    printf("%d ", q[i]);
    if f == (i+1) % que_size;
}
int main() {
    int choice;
    for(;;) {
        printf("1. insert rear in 2. delete front in 3. display in
4. exit \n");
        printf("Enter the choice : ");
        scanf("%d", &choice);
        switch(choice) {
            case 1: printf("Enter the item to be inserted : ");
                scanf("%d", &item);
                insertrear();
                break;
            case 2: item = deletefront();
                if(item == -1)
                    printf("queue is empty \n");
                else
                    printf("item deleted is %d \n", item);
                break;
            case 3: displayq();
                break;
            default: exit(0);
        }
    }
    getch();
    return 0;
}

```

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\circular queue\circular_q.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<process.h>
4 #include<conio.h>
5 #define que_size 3
6 int item, front = 0, rear = -1, q[que_size], count = 0;
7 void insertrear()
8 {
9     if(count == que_size)
10    {
11        printf("queue overflow");
12        return;
13    }
14    rear = (rear + 1) % que_size;
15    q[rear] = item;
16    count++;
17 }
18 int deletefront()
19 {
20     if(count == 0) return -1;
21     item = q[front];
22     front = (front + 1) % que_size;
23     count = count - 1;
24     return item;
25 }
26 void displayq()
27 {
28     int i, f;
29     if(count == 0)
30    {
31        printf("queue is empty");
32        return;
33    }
34    f = front;
35    printf("contents of queue \n");
36    for(i = 0; i <= count; i++)
37    {
38        printf("%d\n", q[f]);
39    }
40 }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Sel: 0 Lines: 69 Length: 1234 Insert Done parsing in 0.906 seconds

Type here to search

11:49 ENG 06-11-2020 6

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
circular_q.cpp
33     }
34     f=front;
35     printf("contents of queue \n");
36     for(i=0;i<=count;i++)
37     {
38         printf(" %d\n", q[f]);
39         f=(f+1)%que_size;
40     }
41 }
42 int main()
43 {
44     int choice;
45     for(;;)
46     {
47         printf("\n1.Insert rear \n2.Delete front \n3.Display \n4.exit \n ");
48         printf("Enter the choice : ");
49         scanf("%d", &choice);
50         switch(choice)
51     {
52         case 1:printf("Enter the item to be inserted : ");
53         scanf("%d", &item);
54         insertrear();
55         break;
56         case 2:item=deletefront();
57         if(item==-1)
58             printf("queue is empty\n");
59         else
60             printf("item deleted is %d \n", item);
61         break;
62         case 3:displayq();
63         break;
64         default:exit(0);
65     }
66 }
67 getch();
68 return 0;
69 }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 1 Sel: 0 Lines: 69 Length: 1234 Insert Done parsing in 0.906 seconds

Type here to search

11:49 ENG 06-11-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\circular queue\circular_q.exe

```
1.Insert rear  
2.Delete front  
3.Display  
4.exit  
Enter the choice : 1  
Enter the item to be inserted :34
```

```
1.Insert rear  
2.Delete front  
3.Display  
4.exit  
Enter the choice : 1  
Enter the item to be inserted :23
```

```
1.Insert rear  
2.Delete front  
3.Display  
4.exit  
Enter the choice : 1  
Enter the item to be inserted :6
```

```
1.Insert rear  
2.Delete front  
3.Display  
4.exit  
Enter the choice : 2  
item deleted is 34
```

```
1.Insert rear  
2.Delete front  
3.Display  
4.exit  
Enter the choice : 3  
contents of queue  
23  
6  
34
```

```
1.Insert rear  
2.Delete front  
3.Display  
4.exit  
Enter the choice : 4
```

```
-----  
Process exited after 39.08 seconds with return value 0  
Press any key to continue . . .
```



Lab program 5:

- 5) Write a program to implement Singly linked list with following operations : a> Create an linked list , b> Insertion of a node at first position, at any position and at end of list . c> Display the contents of linked list .

```
A{ #include <stdio.h>
    #include <conio.h>
    #include <malloc.h>
    #include <process.h>
    struct node
    {
        int info;
        struct node *link;
    };
    typedef struct node *NODE;
    NODE getnode()
    {
        NODE x;
        x = (NODE)malloc(sizeof(struct node));
        if (x == NULL)
        {
            printf(" mem full \n");
            exit(0);
        }
        return x;
    }
    NODE freeNode(NODE x)
    {
        free(x);
        return 0;
    }
    NODE insert_front(NODE first, int item)
    {
        NODE temp;
        temp = getnode();
        temp->info = item;
        temp->link = NULL;
```

```
if (first == NULL) {
    return temp;
}
temp->link = first;
first = temp;
return first;
```

```
} // f
```

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

```
} // f
```

```
void display(NODE first) {
    NODE temp;
    if (first == NULL)
        printf("list empty cannot display items \n");
    for (temp = first; temp != NULL; temp = temp->link) {
        printf("%d \n", temp->info);
    }
}
```

```
} // f
```

```
NODE insert_pos(int item, int pos, NODE first) {
    NODE temp;
    NODE prev, cur;
    int count;
    temp = getnode();
    temp->info = item;
    temp->link = NULL;
```

```
if(first == NULL && pos == -1) {  
    return temp;
```

{

```
if(first == NULL) {  
    printf("invalid position\n");  
    return first;
```

}

```
if(pos == -1) {  
    temp->link = first;  
    return temp;
```

{

```
count = 1;  
prev = NULL;  
cur = first;  
while(cur != NULL && count != pos) {  
    prev = cur;  
    cur = cur->link;  
    count++;
```

{

```
if(count == pos) {  
    prev->link = temp;  
    temp->link = cur;  
    return first;
```

{

```
printf("invalid position\n");  
return first;
```

{

```
int main() {
```

```
int item, choice, pos;  
NODE first = NULL;
```

```
system("cls");
```

```
for(;;) {
```

```
    printf("\n 1: Insert front\n 2: Insert at specified  
position\n 3: Insert rear\n 4: Display list\n 5: Exit
```

in");

~~choice~~;

printf("enter the choice \n");

scanf("%d", &choice);

switch(choice) {

case 1: printf("enter the item at front-end: \n");

scanf("%d", &item);

first = insert_front(first, item);

break;

case 2: printf("enter the item to be inserted: \n");

scanf("%d", &item);

printf("enter the position at which item to be inserted: \n");

scanf("%d", &pos);

first = insert_pos(item, pos, first);

break;

case 3: printf("enter the item at rear-end \n");

scanf("%d", &item);

first = insert_rear(first, item);

break;

case 4: display(first);

break;

default: exit(0);

break;

}

getch();

return 0;

}

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list insert\singly_linked_lists_insertion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<malloc.h>
4 #include<process.h>
5 struct node
6 {
7     int info;
8     struct node *link;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE) malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 int freenode(NODE x)
23 {
24     free(x);
25     return 0;
26 }
27 NODE insert_front(NODE first, int item)
28 {
29     NODE temp;
30     temp=getnode();
31     temp->info=item;
32     temp->link=NULL;
33     if(first==NULL)
34         return temp;
35     temp->link=first;
36     first=temp;
37     return first;
38 }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 18 Sel: 0 Lines: 139 Length: 2287 Insert Done parsing in 0.016 seconds

Search

20:43 05-12-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list insert\singly_linked_lists_insertion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
singly_linked_lists_insertion.cpp

38 L }
39
40 NODE insert_rear(NODE first, int item)
41 {
42     NODE temp, cur;
43     temp = get_node();
44     temp->info = item;
45     temp->link = NULL;
46     if (first == NULL)
47         return temp;
48     cur = first;
49     while (cur->link != NULL)
50         cur = cur->link;
51     cur->link = temp;
52     return first;
53 }
54
55 void display(NODE first)
56 {
57     NODE temp;
58     if (first == NULL)
59         printf("list empty cannot display items\n");
60     for (temp = first; temp != NULL; temp = temp->link)
61     {
62         printf("%d\n", temp->info);
63     }
64
65 NODE insert_pos( int item, int pos, NODE first )
66 {
67     NODE temp;
68     NODE prev, cur;
69     int count;
70     temp = get_node();
71     temp->info = item;
72     temp->link = NULL;
73     if (first == NULL && pos == 1)
74     {
75         return temp;
76     }
77     else
78     {
79         cur = first;
80         prev = NULL;
81         count = 1;
82         while (cur != NULL && count < pos)
83         {
84             prev = cur;
85             cur = cur->link;
86             count++;
87         }
88         if (cur == NULL)
89             temp->link = NULL;
90         else
91             temp->link = cur;
92         if (prev == NULL)
93             first = temp;
94         else
95             prev->link = temp;
96     }
97     return first;
98 }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 18 Sel: 0 Lines: 139 Length: 2287 Insert Done parsing in 0.016 seconds

Search

20:43 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list insert\singly_linked_lists_insertion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
singly_linked_lists_insertion.cpp
75 |     return temp;
76 | }
77 | if(first==NULL)
78 | {
79 |     printf("invalid position \n");
80 |     return first;
81 | }
82 | if(pos==1)
83 | {
84 |     temp->link=first;
85 |     return temp;
86 | }
87 | count=1;
88 | prev=NULL;
89 | cur=first;
90 | while(cur!=NULL && count!=pos)
91 | {
92 |     prev=cur;
93 |     cur=cur->link;
94 |     count++;
95 | }
96 | if(count==pos)
97 | {
98 |     prev->link=temp;
99 |     temp->link=cur;
100 |    return first;
101 | }
102 | printf("invalid position \n");
103 | return first;
104 |
105 | int main()
106 | {
107 |     int item,choice,pos;
108 |     NODE first=NULL;
109 |     system("cls");
110 |     for(;;)
111 |     {
112 |         printf("\n 1:Insert front\n 2:Insert at specified position \n 3:Insert rear\n 4:Display list\n 6:Exit\n");
113 |         choice=getchar();
114 |         switch(choice)
115 |         {
116 |             case '1':
117 |                 insert_at_head(&first);
118 |                 break;
119 |             case '2':
120 |                 insert_at_pos(&first,&item,&pos);
121 |                 break;
122 |             case '3':
123 |                 insert_at_rear(&first,&item);
124 |                 break;
125 |             case '4':
126 |                 display_list(first);
127 |                 break;
128 |             case '6':
129 |                 exit(0);
130 |                 break;
131 |             default:
132 |                 printf("invalid choice\n");
133 |         }
134 |     }
135 | }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 18 Sel: 0 Lines: 139 Length: 2287 Insert Done parsing in 0.016 seconds

Search

20:43 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list insert\singly_linked_lists_insertion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

(globals)

```
singly_linked_lists_insertion.cpp
103     return first;
104 }
105 int main()
106 {
107     int item_choice, pos;
108     NODE first=NULL;
109     system("cls");
110     for(;;)
111     {
112         printf("\n 1:Insert_front\n 2:Insert at specified position \n 3:Insert_rear\n 4:Display_list\n 6:Exit\n");
113         printf("enter the choice\n");
114         scanf ("%d", &choice);
115         switch(choice)
116         {
117             case 1: printf("enter the item at front-end\n");
118             scanf ("%d", &item);
119             first=insert_front(first,item);
120             break;
121             case 2: printf("enter the item to be inserted:\n");
122             scanf ("%d", &item);
123             printf("enter the position at which item to be inserted:\n");
124             scanf ("%d", &pos);
125             first=insert_pos(item,pos,first);
126             break;
127             case 3: printf("enter the item at rear-end\n");
128             scanf ("%d", &item);
129             first=insert_rear(first,item);
130             break;
131             case 4: display(first);
132             break;
133             default: exit(0);
134             break;
135         }
136     }
137     getch();
138     return 0;
139 }
```

Compiler Resources Compile Log Debug Find Results

Line: 1 Col: 18 Sel: 0 Lines: 139 Length: 2287 Insert Done parsing in 0.016 seconds

Search

20:43 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list insert\singly_linked_lists_insertion.exe

```
1:Insert_front  
2:Insert at specified position  
3:Insert_rear  
4:Display_list  
6:Exit
```

enter the choice

1

enter the item at front-end

54

```
1:Insert_front  
2:Insert at specified position  
3:Insert_rear  
4:Display_list  
6:Exit
```

enter the choice

3

enter the item at rear-end

78

```
1:Insert_front  
2:Insert at specified position  
3:Insert_rear  
4:Display_list  
6:Exit
```

enter the choice

2

enter the item to be inserted:

65

enter the position at which item to be inserted:

3

```
1:Insert_front  
2:Insert at specified position  
3:Insert_rear  
4:Display_list  
6:Exit
```

enter the choice

2

enter the item to be inserted:

1

enter the position at which item to be inserted:

2

```
1:Insert_front  
2:Insert at specified position  
3:Insert_rear  
4:Display_list  
6:Exit
```

enter the choice

2

enter the item to be inserted:

1

enter the position at which item to be inserted:

2

```
1:Insert_front  
2:Insert at specified position  
3:Insert_rear  
4:Display_list  
6:Exit
```

enter the choice

2

enter the item to be inserted:

1

enter the position at which item to be inserted:

2



20:43
ENG 05-12-2020 3

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list insert\singly_linked_lists_insertion.exe

78

1:Insert_front
2:Insert at specified position

3:Insert_rear
4:Display_list
6:Exit

enter the choice

2

enter the item to be inserted:

65

enter the position at which item to be inserted:

3

1:Insert_front
2:Insert at specified position

3:Insert_rear
4:Display_list
6:Exit

enter the choice

2

enter the item to be inserted:

1

enter the position at which item to be inserted:

2

1:Insert_front
2:Insert at specified position

3:Insert_rear
4:Display_list
6:Exit

enter the choice

4

54

1

78

65

1:Insert_front
2:Insert at specified position

3:Insert_rear
4:Display_list
6:Exit

enter the choice



Search



20:44
ENG
05-12-2020
3

Ques Program 6 :-

- Ques Write a program implement Singly linked list with following question operation:
- Create a linked list.
 - Deletion of first element, Specified element and last element in the list.
 - Display the contents of the linked list.

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

struct node
{
    int info;
    struct node *link;
};

typedef struct node *NODE;
NODE getnode()
{
    NODE x = (NODE) malloc ( sizeof ( struct . node ) );
    if ( x == NULL )
    {
        printf (" mem full \n ");
        exit ( 0 );
    }
    return x;
}

int freeNode ( NODE x )
{
    free ( x );
    return 0;
}

NODE insertFront ( NODE first, int item )
{
    NODE temp;
    temp = getnode();
    temp->info = item;
    temp->link = NULL;
    if ( first == NULL )
        return temp;
}
```

`temp->link = first;`

`first = temp;`

`return first;`

`}`

`NODE delete_start (NODE first) {`

`NODE temp;`

`if (first == NULL) {`

`printf ("list is empty. cannot delete \n");`

`return first;`

`}`

`NODE delete_start (NODE first) {`

`NODE cur, prev;`

`if (first == NULL) {`

`printf ("list is empty. cannot delete \n");`

`return first;`

`}`

`if (first->link == NULL) {`

`printf ("item deleted is %d \n", first->info);`

`free (first);`

`return NULL;`

`}`

`prev = NULL;`

`cur = first;`

`while (cur->link != NULL) {`

`prev = cur;`

`cur = cur->link;`

`}`

`printf ("item deleted at rear-end is %d ", cur->info);`

`free (cur);`

`prev->link = NULL;`

`return first;`

`}`

`void display (NODE first) {`

`NODE temp;`

`if (first == NULL)`

```
if (printf ("list empty cannot display items\n"),  
    for (temp = first; temp != NULL; temp = temp->link) {  
        printf ("%d\n", temp->info);  
    }  
}
```

```
} // Node delete_pos(int pos, Node first) {
```

```
Node cur;
```

```
Node prev;
```

```
int count;
```

```
if (first == NULL || pos <= 0) {
```

```
printf ("invalid position\n");
```

```
return NULL;
```

```
} // if (pos == 1) {
```

```
cur = first;
```

```
first = first->link;
```

```
freeNode (cur);
```

```
return first;
```

```
prev = NULL;
```

```
cur = first;
```

```
count = 1;
```

```
while (cur != NULL) {
```

```
if (count == pos) break;
```

```
prev = cur;
```

```
cur = cur->link;
```

```
count++;
```

```
} // if (count == pos) {
```

```
printf ("invalid position\n");
```

```
return first;
```

```
} // if (count == pos) {
```

```
printf ("invalid position specified\n");
```

```
return first;
```

```

forev->link = cur->link;
freemode (cur);
return first;
}

```

```

int main() {
    int item, choice, pos;
    NODE first = NULL;
    system("cls");
    for(;;) {
        printf("In 1: Insert-front In 2: Delete-front In 3: Insert-rear In
               4: Delete-rear In 5: Delete at specified position. In 6: Display-list
               In 7: Exit In ");
        printf("enter the choice In ");
        scanf("%d", &choice);
        switch(choice) {
            case 1: printf('enter the item at front-end In');
                      scanf("%d", &item);
                      first = insert_front(first, item);
                      break;
            case 2: first = delete_front(first);
                      break;
            case 3: printf("enter the item at rear-end In");
                      scanf("%d", &item);
                      first = insert_rear(first, item);
                      break;
            case 4: first = delete_rear(first);
                      break;
            case 5: printf("enter the position of the item to be deleted: In");
                      scanf("%d", &pos);
                      first = delete_pos(pos, first);
                      break;
            case 6: display(first);
                      break;
            default: exit(0); break;
        }
    }
}

```

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
[*] singly_linked_lists_deletion.cpp
1 #include <stdio.h>
2 #include <conio.h>
3 #include <malloc.h>
4 #include <process.h>
5 struct node
6 {
7     int info;
8     struct node *link;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE) malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 int freenode(NODE x)
23 {
24     free(x);
25     return 0;
26 }
27 NODE insert_front(NODE first, int item)
28 {
29     NODE temp;
30     temp=getnode();
31     temp->info=item;
32     temp->link=NULL;
33     if(first==NULL)
34         return temp;
35     temp->link=first;
36     first=temp;
37     return first;
}
```

Compiler Resources Compile Log Debug Find Results

Line: 149 Col: 15 Sel: 0 Lines: 182 Length: 3003 Insert Done parsing in 0.015 seconds

Search

21:49 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
[*] singly_linked_lists_deletion.cpp
37 } return first;
38 }
39 NODE delete_front(NODE first)
40 {
41 NODE temp;
42 if(first==NULL)
43 {
44 printf("list is empty cannot delete\n");
45 return first;
46 }
47 temp=first;
48 temp=temp->link;
49 printf("item deleted at front-end is=%d\n", first->info);
50 free(first);
51 return temp;
52 }
53 NODE insert_rear(NODE first, int item)
54 {
55 NODE temp, cur;
56 temp=getnode();
57 temp->info=item;
58 temp->link=NULL;
59 if(first==NULL)
60 return temp;
61 cur=first;
62 while(cur->link!=NULL)
63 cur=cur->link;
64 cur->link=temp;
65 return first;
66 }
67 NODE delete_rear(NODE first)
68 {
69 NODE cur, prev;
70 if(first==NULL)
71 {
72 printf("list is empty cannot delete\n");
73 return first;
```

Compiler Resources Compile Log Debug Find Results

Line: 149 Col: 15 Sel: 0 Lines: 182 Length: 3003 Insert Done parsing in 0.015 seconds

Search

21:49 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
[*] singly_linked_lists_deletion.cpp
73     return first;
74 }
75 if(first->link==NULL)
76 {
77     printf("item deleted is %d\n",first->info);
78     free(first);
79     return NULL;
80 }
81 prev=NULL;
82 cur=first;
83 while(cur->link!=NULL)
84 {
85     prev=cur;
86     cur=cur->link;
87 }
88 printf("item deleted at rear-end is %d",cur->info);
89 free(cur);
90 prev->link=NULL;
91
92 return first;
93 }
94 void display(NODE first)
95 {
96     NODE temp;
97     if(first==NULL)
98         printf("list empty cannot display items\n");
99     for(temp=first;temp!=NULL,temp=temp->link)
100    {
101        printf("%d\n",temp->info);
102    }
103 }
104 NODE delete_pos(int pos, NODE first)
105 {
106     NODE cur;
107     NODE prev;
108     int count;
109     if(first==NULL || pos<=0)
```

Compiler Resources Compile Log Debug Find Results

Line: 149 Col: 15 Sel: 0 Lines: 182 Length: 3003 Insert Done parsing in 0.015 seconds

Search

21:49 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
[*] singly_linked_lists_deletion.cpp
109 if (first==NULL || pos<=0)
110 {
111     printf("invalid position \n");
112     return NULL;
113 }
114 if (pos==1)
115 {
116     cur=first;
117     first=first->link;
118     freenode(cur);
119     return first;
120 }
121 prev=NULL;
122 cur=first;
123 count=1;
124 while(cur!=NULL)
125 {
126     if(count==pos) break;
127     prev=cur;
128     cur=cur->link;
129     count++;
130 }
131 if(count!=pos)
132 {
133     printf("invalid position \n");
134     return first;
135 }
136 if(count!=pos)
137 {
138     printf("invalid position specified \n");
139     return first;
140 }
141 prev->link=cur->link;
142 freenode(cur);
143 return first;
144 }
145 int main()
```

Compiler Resources Compile Log Debug Find Results

Line: 149 Col: 15 Sel: 0 Lines: 182 Length: 3003 Insert Done parsing in 0.015 seconds

Search

21:49 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

(globals)

```
[*] singly_linked_lists_deletion.cpp
143     return first;
144 }
145 int main()
146 {
147     int item_choice, pos;
148     NODE first=NULL;
149     system("cls");
150     for(;;)
151     {
152         printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5.Delete at specified position \n 6:Display_list\n 7:Ex
153         printf("enter the choice\n");
154         scanf ("%d", &choice);
155         switch(choice)
156         {
157             case 1:printf("enter the item at front-end\n");
158             scanf ("%d", &item);
159             first=insert_front(first,item);
160             break;
161             case 2:first=delete_front(first);
162             break;
163             case 3:printf("enter the item at rear-end\n");
164             scanf ("%d", &item);
165             first=insert_rear(first,item);
166             break;
167             case 4:first=delete_rear(first);
168             break;
169             case 5:printf("enter the position of the item to be deleted: \n");
170             |    scanf ("%d", &pos);
171             |    first=delete_pos(pos,first);
172             break;
173
174             case 6:display(first);
175             break;
176             default:exit(0);
177             break;
178         }
179     }
```

Compiler Resources Compile Log Debug Find Results

Line: 149 Col: 15 Sel: 0 Lines: 182 Length: 3003 Insert Done parsing in 0.015 seconds

Search

21:49 ENG 05-12-2020

File Edit Search View Project Execute Tools AStyle Window Help

(globals)

```
[*] singly_linked_lists_deletion.cpp
147 int item_choice, pos;
148 NODE first=NULL;
149 system("cls");
150 for(;;)
151 {
152 printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5.Delete at specified position \n 6:Display_list\n 7:Ex
153 printf(" enter the choice\n");
154 scanf ("%d", &choice);
155 switch(choice)
156 {
157 case 1:printf("enter the item at front-end\n");
158 scanf ("%d", &item);
159 first=insert_front(first,item);
160 break;
161 case 2:first=delete_front(first);
162 break;
163 case 3:printf("enter the item at rear-end\n");
164 scanf ("%d", &item);
165 first=insert_rear(first,item);
166 break;
167 case 4:first=delete_rear(first);
168 break;
169 case 5:printf("enter the position of the item to be deleted: \n");
170 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
171 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
172 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
173 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
174 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
175 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
176 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
177 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
178 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
179 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
180 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
181 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
182 }
```

Compiler Resources Compile Log Debug Find Results

Line: 149 Col: 15 Sel: 0 Lines: 182 Length: 3003 Insert Done parsing in 0.015 seconds

Search

21:49 ENG 05-12-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.exe

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice

1

enter the item at front-end

56

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice

3

enter the item at rear-end

69

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice

1

enter the item at front-end

4

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice

3

enter the item at rear-end

76

```
1:Insert_front
```

Search



21:50 ENG 05-12-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.exe

enter the item at rear-end

76

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice
5

enter the position of the item to be deleted:
3

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice
1

enter the item at front-end
8

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice
3

enter the item at rear-end
9

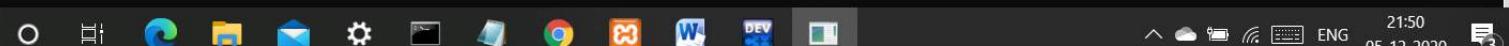
```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit
```

enter the choice
2

item deleted at front-end is=8



Search



21:50 ENG 05-12-2020 3

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\singly linked list delete\singly_linked_lists_deletion.exe

enter the item at rear-end

9

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit  
enter the choice  
2  
item deleted at front-end is=8
```

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit  
enter the choice  
4  
item deleted at rear-end is 9
```

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit  
enter the choice  
6  
4
```

56

76

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
6:Exit  
enter the choice  
7
```

Process exited after 280 seconds with return value 0

Press any key to continue . . .



Lab Program :-

- 1) Write a program implement Single link list with following operations
a) Sort the linked list. b) ~~Reverse~~ Reverse linked list
c) Concatenation of two linked lists.

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

Struct node {
    int info;
    Struct node *link;
};

typedef Struct node *NODE;
NODE getnode() {
    NODE x;
    x = (NODE) malloc(sizeof(Struct node));
    if (x == NULL) {
        printf ("mem full in ");
        exit(0);
    }
    return x;
}

void freenode(NODE x) {
    free(x);
}

NODE insert-front(NODE first, int item) {
    NODE temp;
    temp = getnode();
    temp->info = item;
    temp->link = NULL;
    if (first == NULL) {
        return temp;
    }
    temp->link = first;
}
```

first = temp;
return first;

} NODE IF (NODE second, int item) {
NODE temp;
temp = getnode();
temp->info = item;
temp->link = NULL;
if (second == NULL)
return temp;
temp->link = second;
second = temp;
return second;

} NODE delete-front (NODE first) {
NODE temp;
if (first == NULL){
printf ("list is empty cannot delete \n");
return first;

} temp = first;
temp = temp->link;
printf ("item deleted at front-end is = %d \n", first->info);
free(first);
return temp;

} NODE insert-end (NODE first, int item) {
NODE temp, cur;
temp = getnode();
temp->info = item;
temp->link = NULL;
if (first == NULL)
return temp;
cur = first;

while (cur -> link != NULL)

 cur = cur -> link;

 cur -> link = temp;

return first;

}

NODE LR (NODE second, int item) {

 NODE temp, cur;

 temp = getNode();

 temp -> info = item;

 temp -> link = NULL;

 if (second == NULL)

 return temp;

 cur = second;

 while (cur -> link != NULL)

 cur = cur -> link;

 cur -> link = temp;

 return second;

}

NODE delete_start (NODE first) {

 NODE cur, prev;

 if (first == NULL)

 printf ("list is empty cannot delete \n");

 return first;

}

 if (first -> link == NULL)

 printf ("item deleted is %d \n", first -> info);

 free (first);

 return NULL;

}

 prev = NULL;

 cur = first;

 while (cur -> link != NULL)

 prev = cur;

 cur = cur -> link;

}

```
        printf (" item deleted at rear end is %d ", cur->info );
        free (cur) ;
        forev->link = NULL ;
        return first ;
```

{ NODE insert_pos (int item, int pos, NODE first) {

NODE temp;

NODE forev, cur;

int count;

temp = getnode();

temp->info = item;

temp->link = NULL;

if (first == NULL && pos == 1)

return temp;

if (first == NULL) {

printf (" Invalid pos \n ");

return first;

{ if (pos == 1) {

temp->link = first;

return temp;

{

count = 1;

forev = NULL;

cur = first;

while (cur != NULL && count != pos) {

forev = cur;

cur = cur->link;

count++;

{

if (count == pos) {

forev->link = temp;

temp->link = cur;

return first;

printf ("Invalid position in");
return first;

}

NODE delete_pos (int pos, NODE first) {

NODE cur;

NODE prev;

int count;

if (first == NULL || pos < 0) {

printf ("invalid position\n");

return NULL;

}

if (pos == 1) {

cur = first;

first = first -> link;

freemode (cur);

return first;

}

prev = NULL;

cur = first;

count = 1;

while (cur != NULL) {

if (count == pos)

break;

prev = cur;

cur = cur -> link;

count++;

}

if (count != pos) {

printf ("invalid position\n");

return first;

}

prev -> link = cur -> link;

freemode (cur);

return first;

```
NODE reverse(NODE first){
```

```
    NODE cur, temp;
```

```
    cur = NULL;
```

```
    while (first != NULL) {
```

```
        temp = first;
```

```
        first = first -> link;
```

```
        temp-> link = cur;
```

```
        cur = temp;
```

```
}
```

```
    return cur;
```

```
}
```

```
NODE asc(NODE first){
```

```
    NODE forev = first;
```

```
    NODE cur = NULL;
```

```
    int temp;
```

```
    if (first == NULL) {
```

```
        return 0;
```

```
}
```

```
else {
```

```
    while (forev != NULL) {
```

```
        cur = forev -> link;
```

```
        while (cur != NULL) {
```

```
            if (forev -> info > cur -> info) {
```

```
                temp = forev -> info;
```

```
                forev -> info = cur -> info;
```

```
                cur -> info = temp;
```

```
}
```

```
            cur -> info = cur -> link;
```

```
}
```

```
        forev = forev -> link;
```

```
}
```

```
    return first;
```

```
}
```

```
NODE desc(NODE first){
```

```
    NODE forev = first;
```

```
None cur = NULL;  
int temp;  
if (first == NULL){  
    return 0;  
}  
else{  
    while (forev != NULL){  
        cur = forev->link;  
        while (cur != NULL){  
            if (forev->info < cur->info){  
                temp = forev->info;  
                forev->info = cur->info;  
                cur->info = temp;  
            }  
            cur = cur->link;  
        }  
        forev = forev->link;  
    }  
    return first;  
}
```

NODE concate(NODE first, NODE second){

```
NODE cur;  
if (first == NULL)  
    return second;  
if (second == NULL)  
    return first;  
cur = first;  
while (cur->link != NULL){  
    cur = cur->link;
```

```
}  
cur->link = second;
```

return first;

```
void display(NODE first){  
    None temp;  
    if (first == NULL)
```

```

printf("list empty cannot display items\n");
for(C temp = first; temp != NULL; temp = temp->link) {..}
printf("%d\n", temp->info);

```

83

```

int main() {
    int item, choice, pos, element, option, choice2, Item1, num;
    system("cls");
    Node first = NULL;
    Node second = NULL;
    for(;;) {
        printf("\n 1: insert-front\n 2: delete-front\n 3: Insert-rear\n
        4: delete-rear\n 5: random-position\n 6: reverse\n 7: sort\n 8:
        concat\n 9: display\n 10: Exit\n");
        scanf(" %d", &choice);
        switch(choice) {
            case 1: printf("enter the item at front-end\n");
                scanf("%d", &item);
                first = insert_front(first, item);
                break;
            case 2: first = delete_front(first);
                break;
            case 3: printf("enter the item at rear-end\n");
                scanf("%d", &item);
                first = insert_rear(first, item);
                break;
            case 4: first = delete_rear(first);
                break;
            case 5: printf("press 1 to insert or 2 to delete at any
                        desired position\n");
                scanf("%d", &element);
                if (element == 1) {
                    printf("enter the position to insert\n");
                    scanf("%d", &pos);
                    &

```

```
printf ("enter the item to insert : \n");
scanf ("%d", &item);
first = insert_pos(item, head, first);
if (element == 2) {
    printf ("enter position to delete \n");
    scanf ("%d", &pos);
    first = delete_pos(pos, first);
}
break;
case 6: first = reverse(first);
break;
case 7: printf ("press 1 for ascending : - sort &
2 for descending sort : \n");
scanf ("%d", &option);
if (option == 1)
    first = asc(first);
if (option == 2)
    first = desc(first);
break;
case 8: printf ("Create a second list \n");
printf ("enter the number of elements in second list : \n");
scanf ("%d", &num);
for (int i = 1; i <= num; i++) {
    printf ("In press 1 to insert front & 2 to insert rear in ");
    scanf ("%d", &choice2);
    if (choice2 == 1) {
        printf ("enter the item at the front end \n");
        scanf ("%d", &item1);
    }
    if (choice2 == 2) {
        printf ("enter the item at rear end \n");
        scanf ("%d", &item);
        Second = IR(Second, item1);
    }
}
```

G Proposed Alternate Assessment Tool Plan (if applicable)

Question paper format

One Question to be asked for 20 Marks
One Question to be asked for 20 Marks
and for 20 Marks each

classmate

Date _____
Page _____

first = concat(first, second);
break;

case 9 : display(first);

break;

default : exit(0);

break;

{}

getch();

return 0;

}

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
1 #include <stdio.h>
2 #include <conio.h>
3 #include <malloc.h>
4 #include <stdlib.h>
5 struct node{
6     int info;
7     struct node *link;
8 };
9 typedef struct node *NODE;
10 NODE getnode(){
11     NODE x;
12     x=(NODE) malloc(sizeof(struct node));
13     if(x==NULL){
14         printf("mem full \n");
15         exit(0);
16     }
17     return x;
18 }
19 void freenode(NODE x){
20     free(x);
21 }
22
23 NODE insert_front(NODE first,int item){
24     NODE temp;
25     temp=getnode();
26     temp->info=item;
27     temp->link=NULL;
28     if(first==NULL)
29         return temp;
30     temp->link=first;
31     first=temp;
32     return first;
33 }
34 NODE IF(NODE second,int item){
35     NODE temp;
36     temp=getnode();
37     temp->info=item;
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:18 06-12-2020 ENG

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Linked_list_sort_rev_con.cpp

```
37 |     temp->info=item;
38 |     temp->link=NULL;
39 |     if(second==NULL)
40 |         return temp;
41 |     temp->link==second;
42 |     second=temp;
43 |     return second;
44 |
45 | } NODE delete_front(NODE first){
46 |     NODE temp;
47 |     if(first==NULL){
48 |         printf("list is empty cannot delete \n");
49 |         return first;
50 |     }
51 |     temp=first;
52 |     temp=temp->link;
53 |     printf("item deleted at front-end is= %d \n",first->info);
54 |     free(first);
55 |     return temp;
56 | }
57 | } NODE insert_rear(NODE first,int item){
58 |     NODE temp,cur;
59 |     temp=get_node();
60 |     temp->info=item;
61 |     temp->link=NULL;
62 |     if(first==NULL)
63 |         return temp;
64 |     cur=first;
65 |     while(cur->link!=NULL)
66 |         cur=cur->link;
67 |     cur->link=temp;
68 |     return first;
69 |
70 | } NODE LR(NODE second,int item){
71 |     NODE temp,cur;
72 |     temp=get_node();
73 |     temp->info=item;
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:18 06-12-2020 ENG

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
Linked_list_sort_rev_con.cpp

73 |     temp->info=item;
74 |     temp->link=NULL;
75 |     if(second==NULL)
76 |         return temp;
77 |     cur=second;
78 |     while(cur->link!=NULL)
79 |         cur=cur->link;
80 |         cur->link=temp;
81 |         return second;
82 |
83 ┌ NODE delete_rear(NODE first){
84 |     NODE cur,prev;
85 ┌ if(first==NULL){
86 |         printf("list is empty cannot delete \n ");
87 |         return first;
88 |     }
89 ┌ if(first->link==NULL){
90 |         printf("item deleted is %d \n ",first->info);
91 |         free(first);
92 |         return NULL;
93 |     }
94 |     prev=NULL;
95 |     cur=first;
96 |     while(cur->link!=NULL)
97 ┌ {
98 |         prev=cur;
99 |         cur=cur->link;
100 |    }
101 |    printf("item deleted at rear end is %d",cur->info);
102 |    free(cur);
103 |    prev->link=NULL;
104 |    return first;
105 |}
106 ┌ NODE insert_pos(int item,int pos,NODE first){
107 |     NODE temp;
108 |     NODE prev,cur;
109 |     int count;
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:18 06-12-2020 ENG

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
109 int count;
110 NODE* get_node();
111 NODE* info_item;
112 NODE* link=NULL;
113 if(first==NULL&&pos==1)
114     return first;
115 if(first==NULL){
116     printf("invalid pos \n");
117     return first;
118 }
119 if(pos==1){
120     temp->link=first;
121     return temp;
122 }
123 count=1;
124 prev=NULL;
125 cur=first;
126 while(cur!=NULL&&count!=pos){
127     prev=cur;
128     cur=cur->link;
129     count++;
130 }
131 if(count==pos){
132     prev->link=temp;
133     temp->link=cur;
134     return first;
135 }
136 printf("invalid position \n");
137 return first;
138 }
139 NODE* delete_pos(int pos, NODE* first){
140 NODE* cur;
141 NODE* prev;
142 int count;
143 if(first==NULL||pos<=0){
144     printf("invalid position \n");
145     return NULL;
146 }
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:18 06-12-2020 ENG

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
145     return NULL;
146 }
147 if (pos==1){
148     cur=first;
149     first=first->link;
150     freenode(cur);
151     return first;
152 }
153 prev=NULL;
154 cur=first;
155 count=1;
156 while(cur!=NULL){
157     if (count==pos)
158         break;
159     prev=cur;
160     cur=cur->link;
161     count++;
162 }
163 if (count!=pos){
164     printf("invalid position \n");
165     return first;
166 }
167 if (count!=pos){
168     printf("invalid position specified \n");
169     return first;
170 }
171 prev->link=cur->link;
172 freenode(cur);
173 return first;
174 }
175 NODE reverse(NODE first){
176     NODE cur,temp;
177     cur=NULL;
178     while(first!=NULL){
179         temp=first;
180         first=first->link;
181         temp->link=cur;
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

09:18 06-12-2020 ENG

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
181     temp->link=cur;
182     cur=temp;
183 }
184     return cur;
185 }
186 NODE asc(NODE first){
187     NODE prev=first;
188     NODE cur=NULL;
189     int temp;
190     if(first==NULL){
191         return 0;
192     }
193     else{
194         while(prev!=NULL){
195             cur=prev->link;
196             while(cur!=NULL){
197                 if(prev->info > cur->info){
198                     temp=prev->info;
199                     prev->info=cur->info;
200                     cur->info=temp;
201                 }
202                 cur=cur->link;
203             }
204             prev=prev->link;
205         }
206     }
207     return first;
208 }
209 NODE des(NODE first){
210     NODE prev=first;
211     NODE cur=NULL;
212     int temp;
213     if(first==NULL){
214         return 0;
215     }
216     else{
217         while(prev!=NULL){
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:19 06-12-2020 ENG

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
216     el se{
217     whil e(prev!=NULL){
218         cur=prev->link;
219         whil e(cur!=NULL){
220             if (prev->info < cur->info){
221                 temp=prev->info;
222                 prev->info=cur->info;
223                 cur->info=temp;
224             }
225             cur=cur->link;
226         }
227         prev=prev->link;
228     }
229     retur n first;
230 }
231
232 NODE concat e(NODE first, NODE second){
233     NODE cur;
234     if (first==NULL)
235         retur n second;
236     if (second==NULL)
237         retur n first;
238     cur=first;
239     whil e(cur->link!=NULL){
240         cur=cur->link;
241     }
242     cur->link=second;
243     retur n first;
244 }
245 void display(NODE first){
246     NODE temp;
247     if (first==NULL)
248         printf("list empty cannot display items \n");
249     for (temp=first; temp!=NULL; temp=temp->link){
250         printf("%d \n", temp->info);
251     }
252 }
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:19 06-12-2020 ENG

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
252 L }
253 int main(){
254     int item, choice, pos, element, option, choice2, item1, num;
255     system("cls");
256     NODE first=NULL;
257     NODE second=NULL;
258     for(;;){
259         printf("\n 1: insert_front \n 2: delete_front \n 3: insert_rear \n 4: delete_rear \n 5: random_position \n 6: reverse \n 7: sort \n 8: exit");
260         printf(" enter the choice \n");
261         scanf(" %d", &choice);
262         switch(choice){
263             case 1: printf(" enter the item at front-end \n");
264             scanf(" %d", &item);
265             first=insert_front(first, item);
266             break;
267             case 2: first=delete_front(first);
268             break;
269             case 3: printf(" enter the item at rear-end \n");
270             scanf(" %d", &item);
271             first=insert_rear(first, item);
272             break;
273             case 4: first=delete_rear(first);
274             break;
275             case 5: printf(" press 1 to insert or 2 to delete at any desired position \n");
276             scanf(" %d", &element);
277             if(element==1){
278                 printf(" enter the position to inset \n");
279                 scanf(" %d", &pos);
280                 printf(" enter the item to inset \n");
281                 scanf(" %d", &item);
282                 first=insert_pos(item, pos, first);}
283             if(element==2){
284                 printf(" enter the position to delete \n");
285                 scanf(" %d", &pos);
286                 first=delete_pos(pos, first);}
287             }
288             break;
289         }
290     }
291 }
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:19
ENG 06-12-2020

Linked_list_sort_rev_concat - [Linked_list_sort_rev_concat.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
287 }
288 break;
289 case 6: first=reverse(first);
290     break;
291 case 7: printf("press 1 for ascending sort and 2 for descending sort: \n");
292 scanf("%d", &option);
293 if(option==1)
294 first=asc(first);
295 if(option==2)
296 first=des(first);
297 break;
298 case 8: printf("create a second list \n");
299 printf("enter the number of elements in second list \n");
300 scanf("%d", &num);
301 for(int i=1; i<=num; i++){
302 printf("\n press 1 to insert front and 2 to insert rear \n");
303 scanf("%d", &choice2);
304 if(choice2==1){
305 printf("enter the item at front-end \n");
306 scanf("%d", &item1);
307 }
308 if(choice2==2){
309 printf("enter the item at rear-end \n");
310 scanf("%d", &item1);
311 second=L(second, item1);
312 }
313 }
314 first=concat(first, second);
315 break;
316 case 9: display(first);
317 break;
318 default: exit(0);
319 break;
320 }
321 getch();
322 return 0;
323 }
```

Compiler Resources Compile Log Debug Find Results

Line: 218 Col: 16 Sel: 0 Lines: 323 Length: 5892 Insert Done parsing in 0.063 seconds

Search

09:19 06-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked list sort rev concatenation\Linked_list_sort_rev_concat.exe

```
1:insert_front  
2: delete_front  
3: insert_rear  
4: delete_rear  
5: random_position  
6: reverse  
7: sort  
8: concat  
9: display_list  
10: exit  
enter the choice
```

1

enter the item at front-end

3

```
1:insert_front  
2: delete_front  
3: insert_rear  
4: delete_rear  
5: random_position  
6: reverse  
7: sort  
8: concat  
9: display_list  
10: exit  
enter the choice
```

1

enter the item at front-end

5

```
1:insert_front  
2: delete_front  
3: insert_rear  
4: delete_rear  
5: random_position  
6: reverse  
7: sort  
8: concat  
9: display_list  
10: exit  
enter the choice
```

3

enter the item at rear-end

2

```
1:insert_front  
2: delete_front  
3: insert_rear  
4: delete_rear
```



09:19 ENG 06-12-2020 3

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked list sort rev concatenation\Linked_list_sort_rev_concat.exe

3
enter the item at rear-end
2

1:insert_front
2: delete_front
3: insert_rear
4: delete_rear
5: random_position
6: reverse
7: sort
8: concat
9: display_list
10: exit

enter the choice
3

enter the item at rear-end
4

1:insert_front
2: delete_front
3: insert_rear
4: delete_rear
5: random_position
6: reverse
7: sort
8: concat
9: display_list
10: exit

enter the choice
6

1:insert_front
2: delete_front
3: insert_rear
4: delete_rear
5: random_position
6: reverse
7: sort
8: concat
9: display_list
10: exit

enter the choice
9

4
2
3
5

1:insert_front

Search



09:19 ENG 06-12-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked list sort rev concatenation\Linked_list_sort_rev_concat.exe

```
1:insert_front  
2: delete_front  
3: insert_rear  
4: delete_rear  
5: random_position  
6: reverse  
7: sort  
8: concat  
9: display_list  
10: exit  
enter the choice  
7
```

press 1 for ascending sort and 2 for descending sort:

1

```
1:insert_front  
2: delete_front  
3: insert_rear  
4: delete_rear  
5: random_position  
6: reverse  
7: sort  
8: concat  
9: display_list  
10: exit  
enter the choice  
9
```

2

3

4

5

```
1:insert_front  
2: delete_front  
3: insert_rear  
4: delete_rear  
5: random_position  
6: reverse  
7: sort  
8: concat  
9: display_list  
10: exit  
enter the choice  
8
```

create a second list

enter the number of elements in second list

3

press 1 to insert front and 2 to insert rear



Search



09:20 ENG 06-12-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked list sort rev concatenation\Linked_list_sort_rev_concat.exe

```
1
enter the item at front-end
33

press 1 to insert front and 2 to insert rear
2
enter the item at rear-end
22

press 1 to insert front and 2 to insert rear
1
enter the item at front-end
11

1:insert_front
2: delete_front
3: insert_rear
4: delete_rear
5: random_position
6: reverse
7: sort
8: concate
9: display_list
10: exit
enter the choice
7
press 1 for ascending sort and 2 for descending sort:
2

1:insert_front
2: delete_front
3: insert_rear
4: delete_rear
5: random_position
6: reverse
7: sort
8: concate
9: display_list
10: exit
enter the choice
9
22
5
4
3
2

1:insert_front
2: delete_front
3: insert_rear
```



09:20 ENG 06-12-2020 3

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked list sort rev concatenation\Linked_list_sort_rev_concat.exe

```
10: exit
enter the choice
7
press 1 for ascending sort and 2 for descending sort:
2

1:insert_front
2: delete_front
3: insert_rear
4: delete_rear
5: random_position
6: reverse
7: sort
8: concat
9: display_list
10: exit
enter the choice
9
22
5
4
3
2

1:insert_front
2: delete_front
3: insert_rear
4: delete_rear
5: random_position
6: reverse
7: sort
8: concat
9: display_list
10: exit
enter the choice
10
```

Process exited after 54.1 seconds with return value 0
Press any key to continue . . .



Lab Program 8:-
8) Write a program to implement Stack & Queue using
linked representation.

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<process.h>
Struct node
{
    int info;
    struct node *link;
};
typedef struct struct node *NODE;
NODE getnode()
{
    NODE x;
    x = (NODE) malloc(sizeof(struct node));
    if (x == NULL)
    {
        printf(" mem full \n");
        exit(0);
    }
    return x;
}
void freenode(NODE x)
{
    free(x);
}
NODE insert_rear(NODE first, int item)
{
    NODE temp, cur;
    temp = getnode();
    temp->info = item;
```

```
temp → link = NULL;  
if (first == NULL)  
    return temp;  
cur = first;  
while (cur → link != NULL)  
    cur = cur → link;  
cur → link = temp;  
return first;
```

{

```
NODE delete_front(NODE first){
```

```
NODE temp;
```

```
if (first == NULL) {
```

```
    printf("list is empty cannot delete \n");
```

```
    return first;
```

{

```
temp = first;
```

```
temp = temp → link;
```

```
printf("item deleted at front-end is = %d \n", first → info);
```

```
free(first);
```

```
return temp;
```

{

```
void display(NODE first){
```

```
NODE temp;
```

```
if (first == NULL) {
```

```
    printf("list empty cannot display items \n");
```

```
for (temp = first; temp != NULL; temp = temp → link).
```

{

```
    printf("%d \n", temp → info);
```

{

```
NODE insert_front(NODE first, int item){
```

```
NODE temp;
```

```
temp = get-node();
```

```
temp → info = item;
```

```
temp → link = NULL;
```

```
if (first == NULL)
    return temp;
temp->link = first;
first = temp;
return first;
```

}

```
NODE delete_at_front (NODE first) {
```

```
NODE temp;
if (first == NULL) {
    printf ("stack is empty cannot delete \n");
    return first;
}
```

```
temp = first;
temp = temp->link;
printf ("item deleted at front-end is = %d \n", first->info);
free (first);
return temp;
```

}

```
void display_s (NODE first) {
```

```
NODE temp;
if (first == NULL)
```

```
printf ("stack empty cannot display items \n");
for (temp = first; temp != NULL; temp = temp->link)
```

```
{
```

```
int main () {
```

```
int item, choice, flag;
```

```
NODE first = NULL;
```

```
system ("cls");
```

```
for (;;) {
```

```
printf ("In Queue operation :\n 1: insert rear\n 2: delete front
```

```
 3: display-list (queue)\n 4: stack operations\n
```

```
 5: insert front\n 6: delete front\n 7: display-list (stack)\n
```

```
7: Exit \n");
printf(" enter the choice \n");
scanf("%d",&choice);
switch(choice)
{
    case 1: printf("enter the item at rear-end \n");
        scanf("%d",&item);
        first = insert_rear(first,item);
        break;
    case 2: first = delete_rear(first);
        break;
    case 3: display(first);
        break;
    case 4: printf(" enter the item at front-end \n");
        scanf("%d",&item);
        first = insert_front(first,item);
        break;
    case 5: first = delete_front(first);
        break;
    case 6: display_s(first);
        break;
    default: exit(0);
        break;
}
getch();
return 0;
}
```

Linked_lists_stacks_queues - [Linked_lists_stacks_queues.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
linked_lists_queue_stack.cpp
```

```
1 #include <stdio.h>
2 #include <conio.h>
3 #include <malloc.h>
4 #include <process.h>
5 struct node
6 {
7     int info;
8     struct node *link;
9 };
10 typedef struct node *NODE;
11 NODE getnode()
12 {
13     NODE x;
14     x=(NODE) malloc(sizeof(struct node));
15     if(x==NULL)
16     {
17         printf("mem full\n");
18         exit(0);
19     }
20     return x;
21 }
22 void freenode(NODE x)
23 {
24     free(x);
25 }
26 NODE insert_rear(NODE first, int item)
27 {
28     NODE temp, cur;
29     temp=getnode();
30     temp->info=item;
31     temp->link=NULL;
32     if(first==NULL)
33         return temp;
34     cur=first;
35     while(cur->link!=NULL)
36         cur=cur->link;
37     cur->link=temp;
```

Compiler Resources Compile Log Debug Find Results

Line: 97 Col: 15 Sel: 0 Lines: 134 Length: 2388 Insert Done parsing in 0.031 seconds

public int __cdecl printf(const char * __restrict__ _Format, ...)

22:49 ENG 05-12-2020

Linked_lists_stacks_queues - [Linked_lists_stacks_queues.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
linked_lists_queue_stack.cpp
```

```
36 | cur=cur->link;
37 | cur->link=temp;
38 | return first;
39 |
40 NODE delete_front(NODE first)
41 {
42 NODE temp;
43 if(first==NULL)
44 {
45 printf("list is empty cannot delete\n");
46 return first;
47 }
48 temp=first;
49 temp=temp->link;
50 printf("item deleted at front-end is=%d\n", first->info);
51 free(first);
52 return temp;
53 }
54 void display(NODE first)
55 {
56 NODE temp;
57 if(first==NULL)
58 printf("list empty cannot display items\n");
59 for(temp=first; temp!=NULL; temp=temp->link)
60 {
61 printf ("%d \n", temp->info);
62 }
63 }
64 NODE insert_front(NODE first, int item)
65 {
66 NODE temp;
67 temp=get_node();
68 temp->info=item;
69 temp->link=NULL;
70 if(first==NULL)
71 return temp;
72 temp->link=first;
```

Compiler Resources Compile Log Debug Find Results

Line: 97 Col: 15 Sel: 0 Lines: 134 Length: 2388 Insert Done parsing in 0.031 seconds

public int __cdecl printf(const char * __restrict__ _Format, ...)

22:49 ENG 05-12-2020

Linked_lists_stacks_queues - [Linked_lists_stacks_queues.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
linked_lists_queue_stack.cpp

73     first=temp;
74     return first;
75 }
76 NODE delete_front_s(NODE first)
77 {
78     NODE temp;
79     if(first==NULL)
80     {
81         printf("stack is empty cannot delete\n");
82         return first;
83     }
84     temp=first;
85     temp=temp->link;
86     printf("item deleted at front-end is=%d\n", first->info);
87     free(first);
88     return temp;
89 }
90 void display_s(NODE first)
91 {
92     NODE temp;
93     if(first==NULL)
94     printf("stack empty cannot display items\n");
95     for(temp=first; temp!=NULL; temp=temp->link)
96     {
97         printf ("%d\n", temp->info);
98     }
99 }
100 int main()
101 {
102     int item choice, pos;
103     NODE first=NULL;
104     system("cls");
105     for(;;)
106     {
107         printf("\n Queue operations :\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list(Queue)\n \n Stack operations \n 4:Insert_front\n 5:
108         printf("enter the choice \n");
109         scanf (" %d", &choice);
```

Compiler Resources Compile Log Debug Find Results

Line: 97 Col: 15 Sel: 0 Lines: 134 Length: 2388 Insert Done parsing in 0.031 seconds

Search

22:49 05-12-2020 ENG

Linked_lists_stacks_queues - [Linked_lists_stacks_queues.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
linked_lists_stack.cpp
public int __cdecl printf(const char * __restrict__ _Format, ...)
99 L }
100 int main()
101 {
102     int item_choice, pos;
103     NODE first=NULL;
104     system("cls");
105     for(;;)
106     {
107         printf("\n Queue operations :\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list(Queue)\n\n Stack operations \n 4:Insert_front\n 5:
108         printf("enter the choice \n");
109         scanf ("%d", &choice);
110         switch(choice)
111         {
112             case 1: printf("enter the item at rear-end\n");
113             scanf ("%d", &item);
114             first=insert_rear(first,item);
115             break;
116             case 2: first=delete_front(first);
117             break;
118             case 3: display(first);
119             break;
120             case 4: printf("enter the item at front-end\n");
121             scanf ("%d", &item);
122             first=insert_front(first,item);
123             break;
124             case 5: first=delete_front_s(first);
125             break;
126             case 6: display_s(first);
127             break;
128             default: exit(0);
129             break;
130         }
131     }
132     getch();
133     return 0;
134 }
```

Compiler Resources Compile Log Debug Find Results

Line: 97 Col: 15 Sel: 0 Lines: 134 Length: 2388 Insert Done parsing in 0.031 seconds

Search

22:50 05-12-2020 ENG

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked_list_stack_queues\Linked_lists_stacks_queues.exe

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

enter the choice

1
enter the item at rear-end
34

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

enter the choice

1
enter the item at rear-end
54

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

enter the choice

2
item deleted at front-end is=34

```
Queue operations :  
1:Insert_rear
```

Windows Search



22:50
ENG
05-12-2020



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked_list_stack_queues\Linked_lists_stacks_queues.exe

```
enter the choice  
2  
item deleted at front-end is=34
```

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

```
enter the choice
```

```
3  
54
```

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

```
enter the choice
```

```
4  
enter the item at front-end  
6
```

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

```
enter the choice
```

```
4  
enter the item at front-end  
7
```



22:50
ENG 05-12-2020 3

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked_list_stack_queues\Linked_lists_stacks_queues.exe

7:Exit

enter the choice

4

enter the item at front-end

7

Queue operations :

1:Insert_rear

2:Delete_front

3:Display_list(Queue)

Stack operations

4:Insert_front

5: Delete front

6:Display_list(Stack)

7:Exit

enter the choice

4

enter the item at front-end

8

Queue operations :

1:Insert_rear

2:Delete_front

3:Display_list(Queue)

Stack operations

4:Insert_front

5: Delete front

6:Display_list(Stack)

7:Exit

enter the choice

5

item deleted at front-end is=8

Queue operations :

1:Insert_rear

2:Delete_front

3:Display_list(Queue)

Stack operations

4:Insert_front

5: Delete front

6:Display_list(Stack)

7:Exit

enter the choice

7:Exit



22:50
ENG 05-12-2020 3

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\linked_list_stack_queues\Linked_lists_stacks_queues.exe

2:Delete_front
3:Display_list(Queue)

Stack operations
4:Insert_front
5: Delete_front
6:Display_list(Stack)
7:Exit

enter the choice

5

item deleted at front-end is=8

Queue operations :
1:Insert_rear
2:Delete_front
3:Display_list(Queue)

Stack operations
4:Insert_front
5: Delete_front
6:Display_list(Stack)
7:Exit

enter the choice

6

7

6

54

Queue operations :
1:Insert_rear
2:Delete_front
3:Display_list(Queue)

Stack operations
4:Insert_front
5: Delete_front
6:Display_list(Stack)
7:Exit

enter the choice

7

Process exited after 78.14 seconds with return value 0

Press any key to continue . . .



Search



22:50
ENG
05-12-2020



Kab Program - 9

q) Write a program to implement doubly linked list with following operations:

i) Insert front iv) Delete rear

ii) Insert rear v) Display

iii) Delete front vi) Insert after/ before key node

vii) Search

viii) Delete duplicate

A) #include < stdio.h >

#include < conio.h >

#include < process.h >

#include < std.lib.h >

#include < malloc.h >

struct node {

int info;

struct node *llink;

struct node *glink;

};

typedef struct node *NODE;

NODE getnode()

{ NODE x;

x = (NODE) malloc (sizeof (struct node));

if (x == NULL)

printf ("mem full");

exit(0);

}

return x;

}

void freenode (NODE x)

free(x);

}

NODE insert_front (int item, NODE head)

NODE temp, cur;

```
temp = getnode();
temp->info = item;
cur = head->glink;
head->glink = temp;
temp->llink = head;
temp->glink = cur;
cur->llink = temp;
return head;
```

{

```
NODE insert_gear(int item, NODE head){
```

```
NODE temp, cur;
temp = getnode();
temp->info = item;
cur = head->llink;
head->llink = temp;
temp->glink = head;
temp->llink = cur;
cur->glink = temp;
return head;
```

{

```
NODE delete_front(NODE head){
```

```
NODE cur, next;
if(head->glink == head){
    printf("dq empty\n");
    return head;
}
```

{

```
cur = head->glink;
```

```
next = cur->glink;
```

```
head->glink = next;
```

```
next->llink = head;
```

```
printf("the node deleted is %d", cur->info);
```

```
freenode(cur);
```

```
return head;
```

{

```
NODE delete_near(NODE head){
```

```
    NODE cur, prev;
```

```
    if(head->glink == head){
```

```
        printf(" dq empty \n");
```

```
        return head;
```

```
}
```

```
    cur = head->llink;
```

```
    prev = cur->llink;
```

```
    head->llink = prev;
```

```
    prev->glink = head;
```

```
    printf(" the node deleted is r. d ", cur->info);
```

```
    freeNode(cur);
```

```
    return head;
```

```
}
```

```
void display(NODE head){
```

```
    NODE temp;
```

```
    if(head->glink == head){
```

```
        printf(" dq empty \n");
```

```
        return;
```

```
}
```

```
    printf(" contents of dq \n");
```

```
    temp = head->glink;
```

```
    while(temp != head){
```

```
        printf(" r. d ", temp->info);
```

```
        temp = temp->glink;
```

```
}
```

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

```
}
```

```
NODE delete_all_key(int item, NODE head){
```

```
    NODE forev, cur, next;
```

```
    int count;
```

```
    if(head->glink == head){
```

```
        printf(" List Empty \n");
```

```
        return head;
```

```
}
```

```
count = 0;  
cur = head -> glink;  
curr = cur -> glink;  
while (cur != head) {  
    if (item != cur -> info) {  
        cur = cur -> glink;  
    } else {  
        count++;  
        prev = cur -> llink;  
        next = cur -> rlink;  
        prev -> rlink = next;  
        next -> llink = prev;  
        freeNode (cur);  
        cur = next;  
    }  
}
```

```
if (count == 0)  
    printf ("key not found in");  
else  
    printf ("key found at %d positions and are deleted in", count);  
return head;
```

```
3  
NODE insert_left (int item, NODE head) {  
    NODE temp, cur, prev;  
    if (head -> glink == head) {  
        printf ("Empty\n");  
        return head;  
    }
```

```
    cur = head -> glink;  
    while (cur != head) {  
        if (item == cur -> info) break;  
        cur = cur -> glink;  
    }  
    if (cur == head) {  
        printf ("key not found in");  
    }
```

return head;
}

prev = cur -> llink;
printf("Enter towards left of %d: ", item);
temp = getnode();
scanf("%d", &temp->info);
prev -> llink = temp;
temp -> llink = prev;
cur -> llink = temp;
temp -> rlink = cur;
return head;

{
NODE insert_right(int item, NODE head){
NODE temp, cur, prev;
if(head -> llink == head){
printf("Empty \n");
return head;

}
cur = head -> llink;
while(cur != head){
if(item == cur -> info), break;
cur = cur -> llink;

}
if(cur == head){
printf("key not found \n");
return head;

}
prev = cur -> rlink;
printf("Enter towards right of %d: ", item);
temp = getnode();
scanf("%d", &temp->info);
prev -> rlink = temp;
temp -> rlink = prev;
cur -> rlink = temp;

```

    tonph->link = aor;
    return head;
}

NODE search(NODE head) {
    NODE temp = head->link;
    int count = 0, key, flag = 0;
    printf("Enter the key : ");
    scanf("%d", &key);
    while (temp != head) {
        count++;
        if (temp->info == key) {
            flag = 1;
            printf("key %d found in position %d", key, count);
        }
        temp = temp->link;
    }
    if (flag == 0)
        printf("key is not found in list");
    return head;
}

int main() {
    NODE head, last;
    int item, choice, option;
    head = getnode();
    head->link = head;
    head->llink = head;
    system("cls");
    for (;;) {
        printf("1: insert front 2: insert rear 3: delete front in\n"
               "4: delete rear 5: display 6: Delete repeating occurrences in 7:\n"
               "search 8: Insert node after/before key node 9: Exit\n");
        printf("enter the choice in ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                item = getitem();
                insertfront(item);
                break;
            case 2:
                item = getitem();
                insertrear(item);
                break;
            case 3:
                deletefront();
                break;
            case 4:
                deleterear();
                break;
            case 5:
                display();
                break;
            case 6:
                delete();
                break;
            case 7:
                search();
                break;
            case 8:
                item = getitem();
                pos = getpos();
                insertafterbefore(item, pos);
                break;
            case 9:
                exit(0);
                break;
            default:
                printf("Wrong choice");
        }
    }
}

```

switch (choice) {

case 1: printf (" enter the item at front end \n"),
scanf ("%d", &item);
last = insert_front (item, head);
break;

case 2: printf (" enter the item at rear end \n"),
scanf ("%d", &item);
last = insert_rear (item, head);
break;

case 3: last = delete_front (head),
break;

case 4: last = delete_rear (head),
break;

case 5: display (head),
break;

case 6: printf (" enter the item \n"),
scanf ("%d", &item);
last = delete_all_key (item, head);
break;

case 8: printf (" press 1 : for insert behind 2 : for insert after),
scanf ("%d", &option);
if (option == 1) {
printf (" enter # key node \n"),
scanf ("%d", &item);
last = insert_left_pos (item, head); }
else if (option == 2) {
printf (" enter key node \n"),
scanf ("%d", &item);
last = insert_right_pos (item, head); }
break;
}

case 7: search (head),
break;

default: exit(0);

```
    }  
    getch();  
    return (0);  
}
```



TDM-GCC 4.9.2 64-bit Release

(globals)

```
1 #include<stdio.h>
2 #include<conio.h>
3 #include<process.h>
4 #include<stdlib.h>
5 #include <malloc.h>
6 struct node
7 {
8     int info;
9     struct node *llink;
10    struct node *rlink;
11 };
12 typedef struct node *NODE;
13
14
15 NODE getnode()
16 {
17     NODE x;
18     x=(NODE)malloc(sizeof(struct node));
19     if(x==NULL)
20     {
21         printf("mem full\n");
22         exit(0);
23     }
24     return x;
25 }
26
27
28 void freenode(NODE x)
29 {
30     free(x);
31 }
32
33
34 NODE insert_front(int item, NODE head)
35 {
36     NODE temp, cur;
37     temp=getnode();
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert

Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



(globals)

doubly_linked_lists.cpp

```
37 |     temp=get_node();
38 |     temp->info=item;
39 |     cur=head->rlink;
40 |     head->rlink=temp;
41 |     temp->llink=head;
42 |     temp->rlink=cur;
43 |     cur->llink=temp;
44 |     return head;
45 |
46 |
47
48 NODE di nsert _rear(int item, NODE head)
49 {
50     NODE temp, cur;
51     temp=get_node();
52     temp->info=item;
53     cur=head->llink;
54     head->llink=temp;
55     temp->rlink=head;
56     temp->llink=cur;
57     cur->rlink=temp;
58     return head;
59 }
60
61
62 NODE dde let e_fron t(NODE head)
63 {
64     NODE cur, next;
65     if(head->rlink==head)
66     {
67         printf("dq empty\n");
68         return head;
69     }
70     cur=head->rlink;
71     next=cur->rlink;
72     head->rlink=next;
73     next->llink=head;
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035

Insert Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



(globals)

```
doubly_linked_lists.cpp

72     head->rlink=next;
73     next->llink=head;
74     printf("the node deleted is %d", cur->info);
75     freenode(cur);
76     return head;
77 }
78
79
80 NODE ddelete_rear(NODE head)
81 {
82     NODE cur, prev;
83     if(head->rlink==head)
84     {
85         printf("dq empty\n");
86         return head;
87     }
88     cur=head->llink;
89     prev=cur->llink;
90     head->llink=prev;
91     prev->rlink=head;
92     printf("the node deleted is %d", cur->info);
93     freenode(cur);
94     return head;
95 }
96
97
98 void display(NODE head)
99 {
100    NODE temp;
101    if(head->rlink==head)
102    {
103        printf("dq empty\n");
104        return;
105    }
106    printf("contents of dq\n");
107    temp=head->rlink;
108    while(temp!=head)
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



(globals)

```
109 {
110     printf ("%d ", temp->info);
111     temp=temp->rlink;
112 }
113 printf ("\n");
114 }
115
116
117 NODE delete_all_key(int item, NODE head)
118 {
119     NODE prev, cur, next;
120     int count;
121     if (head->rlink==head)
122     {
123         printf ("List Empty");
124         return head;
125     }
126     count=0;
127     cur=head->rlink;
128     cur=cur->rlink;
129     while (cur!=head)
130     {
131         if (item==cur->info)
132             cur=cur->rlink;
133         else
134         {
135             count++;
136             prev=cur->llink;
137             next=cur->rlink;
138             prev->rlink=next;
139             next->llink=prev;
140             freenode(cur);
141             cur=next;
142         }
143     }
144     if (count==0)
145         printf ("key not found");
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



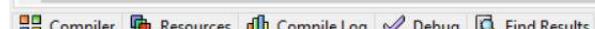
Type here to search



18:29 ENG 15-12-2020 4



```
doubly_linked_lists.cpp
146     else
147     printf("key found at %d positions and are deleted\n", count);
148
149     return head;
150
151
152 }
153 NODE insert_left pos(int item, NODE head){
154     NODE temp, cur, prev;
155     if (head->rlink==head){
156         printf("Empty \n");
157         return head;
158     }
159     cur=head->rlink;
160     while(cur!=head){
161         if(item==cur->info)break;
162         cur=cur->rlink;
163     }
164     if (cur==head){
165         printf("key nt found \n");
166         return head;
167     }
168     prev=cur->llink;
169     printf("Enter towards left of %d:", item);
170     temp=get node();
171     scanf (" %d", &temp->info);
172     prev->rlink=temp;
173     temp->llink=prev;
174     cur->llink=temp;
175     temp->rlink=cur;
176     return head;
177 }
178 NODE insert_right pos(int item, NODE head){
179     NODE temp, cur, prev;
180     if (head->llink==head){
181         printf("Empty \n");
182         return head;
183     }
```





TDM-GCC 4.9.2 64-bit Release

(globals)

```
152 L }
153 NODE insert_left pos(int item, NODE head){
154     NODE temp, cur, prev;
155     if(head->rlink==head){
156         printf("Empty \n");
157         return head;
158     }
159     cur=head->rlink;
160     while(cur!=head){
161         if(item==cur->info)break;
162         cur=cur->rlink;
163     }
164     if(cur==head){
165         printf("key not found \n");
166         return head;
167     }
168     prev=cur->llink;
169     printf("Enter towards left of %d:", item);
170     temp=getnode();
171     scanf("%d", &temp->info);
172     prev->rlink=temp;
173     temp->llink=prev;
174     cur->llink=temp;
175     temp->rlink=cur;
176     return head;
177 NODE insert_right pos(int item, NODE head){
178     NODE temp, cur, prev;
179     if(head->llink==head){
180         printf("Empty \n");
181         return head;
182     }
183     cur=head->llink;
184     while(cur!=head){
185         if(item==cur->info)break;
186         cur=cur->llink;
187     }
188     if(cur==head){
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035

Insert Done parsing in 0.016 seconds



Type here to search



18:29 ENG 15-12-2020 4



(globals)

```
187 }
188 if (cur == head){
189     printf("key not found \n");
190     return head;
191 }
192 prev=cur->rlink;
193 printf("Enter towards right of %d: ", item);
194 temp=getnode();
195 scanf("%d", &temp->info);
196 prev->llink=temp;
197 temp->rlink=prev;
198 cur->rlink=temp;
199 temp->llink=cur;
200 return head; }

201
202
203 NODE search( NODE head)
204 {
205     NODE temp=head->rlink;
206     int count=0, key, flag=0;
207     printf("Enter the key : ");
208     scanf("%d", &key);
209     while(temp!=head)
210     {
211         count++;
212         if (temp->info==key)
213         {
214             flag=1;
215             printf("key %d found in position %d", key, count);
216             temp=temp->rlink;
217         }
218         if (flag==0)
219         {
220             printf("Key is not found in list");
221         }
222     }
223 }
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



Type here to search



18:30 ENG 15-12-2020 4



(globals)

```
doubly_linked_lists.cpp
223 L }
224
225
226 int main()
227 {
228     NODE head, last;
229     int item, choice, option;
230     head=get_node();
231     head->rlink=head;
232     head->llink=head;
233     system("cls");
234     for(;;)
235     {
236         printf("\n1: insert front\n2: insert rear\n3: delete front\n4: delete rear\n5: display\n6: Delete repeating occurrences\n7: search\n 8:");
237         printf(" enter the choice\n");
238         scanf("%d", &choice);
239         switch(choice)
240         {
241             case 1: printf("enter the item at front endl");
242             scanf("%d", &item);
243             last=dinsert_front(item,head);
244             break;
245             case 2: printf("enter the item at rear endl");
246             scanf("%d", &item);
247             last=dinsert_rear(item,head);
248             break;
249             case 3: last=ddel ete_front(head);
250             break;
251             case 4: last=ddel ete_rear(head);
252             break;
253             case 5: display(head);
254             break;
255             case 6: printf("enter the item \n");
256             scanf(" %d", &item);
257             last=del ete_all_key(item,head);
258             break;
259             case 8: printf(" press 1:for insert behind    2: for insert after");
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



File Edit Search View Project Execute Tools AStyle Window Help

(globals)

```
doubly_linked_lists.cpp
233 | system("cls");
234 | for(;;)
235 | {
236 |     printf("\n1: insert front\n2: insert rear\n3: delete front\n4: delete rear\n5: display\n6: Delete repeating occurrences\n7: search\n 8:
237 |     printf("enter the choice\n");
238 |     scanf("%d", &choice);
239 |     switch(choice)
240 |     {
241 |         case 1: printf("enter the item at front end\n");
242 |             scanf("%d", &item);
243 |             last=dinsert_front(item,head);
244 |             break;
245 |         case 2: printf("enter the item at rear end\n");
246 |             scanf("%d", &item);
247 |             last=dinsert_rear(item,head);
248 |             break;
249 |         case 3: last=ddel ete_front(head);
250 |             break;
251 |         case 4: last=ddel ete_rear(head);
252 |             break;
253 |         case 5: display(head);
254 |             break;
255 |         case 6: printf("enter the item \n");
256 |             scanf("%d", &item);
257 |             last=del ete_all_key(item,head);
258 |             break;
259 |         case 8: printf("press 1:for insert behind      2: for insert after");
260 |             scanf("%d", &option);
261 |             if(option==1){
262 |                 printf("enter key node\n");
263 |                 scanf("%d", &item);
264 |                 last=insert_left_pos(item,head); }
265 |             else if(option==2){
266 |                 printf("enter key node\n");
267 |                 scanf("%d", &item);
268 |                 last=insert_right_pos(item,head);
269 |             break;
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds

Type here to search

18:30 15-12-2020 ENG 4



(globals)

```
doubly_linked_lists.cpp
244     break;
245     case 2: printf("enter the item at rear end\n");
246     scanf("%d", &item);
247     last = insert_rear(item, head);
248     break;
249     case 3: last = delete_front(head);
250     break;
251     case 4: last = delete_rear(head);
252     break;
253     case 5: display(head);
254     break;
255     case 6: printf("enter the item \n");
256     scanf("%d", &item);
257     last = delete_all_key(item, head);
258     break;
259     case 8: printf("press 1:for insert behind    2: for insert after");
260     scanf("%d", &option);
261     if(option==1){
262     printf("enter key node\n");
263     scanf("%d", &item);
264     last = insert_left_pos(item, head); }
265     else if(option==2){
266     printf("enter key node\n");
267     scanf("%d", &item);
268     last = insert_right_pos(item, head);
269     break;
270     }
271     case 7: search(head);
272     break;
273     default: exit(0);
274   }
275   getch();
276   return(0);
277 }
```

Compiler Resources Compile Log Debug Find Results

Line: 126 Col: 8 Sel: 0 Lines: 279 Length: 5035 Insert Done parsing in 0.016 seconds



Type here to search



18:30 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
34

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
55

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
67

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
34

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
55

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
67

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
67 55 34 9 13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
7
Enter the key :55
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
7
Enter the key :55
key 55 found in position 2
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
33

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
1

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
1 67 55 34 9 13 33

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
3
the node deleted is 1
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
4
the node deleted is 33
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
67 55 34 9 13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6>Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
9
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
2
enter the item at rear end
9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
1
enter the item at front end
9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
9 67 55 34 9 13 9

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
6
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
9:Exit  
enter the choice  
6  
enter the item  
9  
key found at 2 positions and are deleted
```

```
1:insert front  
2:insert rear  
3:delete front  
4:delete rear  
5:display  
6:Delete repeating occurences  
7:search  
8:Inserting node before/after key node  
9:Exit  
enter the choice
```

```
5  
contents of dq  
9 67 55 34 13
```

```
1:insert front  
2:insert rear  
3:delete front  
4:delete rear  
5:display  
6:Delete repeating occurences  
7:search  
8:Inserting node before/after key node  
9:Exit  
enter the choice
```

```
8  
press 1:for insert behind 2: for insert after1  
enter key node  
67
```

```
Enter towards left of 67:12
```

```
Enter the key :12
```

```
key 12 found in position 2
```

```
1:insert front  
2:insert rear  
3:delete front  
4:delete rear  
5:display  
6:Delete repeating occurences  
7:search  
8:Inserting node before/after key node  
9:Exit  
enter the choice
```

```
5  
contents of dq
```

```
 Type here to search
```



18:33 ENG 15-12-2020 4

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\doubly linked list\doubly_linked_lists.exe

```
enter the choice
8
press 1:for insert behind    2: for insert after1
enter key node
67
Enter towards left of 67:12
Enter the key :12
key 12 found in position 2
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
5
contents of dq
9 12 67 55 34 13

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Delete repeating occurences
7:search
8:Inserting node before/after key node
9:Exit
enter the choice
9
```

Process exited after 150.6 seconds with return value 0
Press any key to continue . . .



Lab Program - 10

10) Write a program

a) To construct a binary search tree.

b) To traverse the tree using all methods i.e., in-order, pre-order & post-order.

c) To display the elements in the tree.

A. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct node {

int info;

struct node *llink;

struct node *rlink;

}

typedef struct node *NODE;

NODE getnode()

NODE x;

x = (NODE) malloc(sizeof(struct node));

if(x == NULL){

printf("Memory not available \n");

exit(0);

}

return x;

}

void freenode(NODE x){

free(x);

}

NODE insort(int item, NODE root){

NODE temp, cur, prev;

char direction[10];

int i;

temp = getnode();

```
temp->info = item;
temp->llink = NULL;
temp->rlink = NULL;
if (root == NULL)
    return temp;
printf("Give direction to insert \n");
scanf("Y.S", direction);
prev = NULL;
cur = root;
for (i = 0; i < strlen(direction) && cur != NULL; i++) {
    prev = cur;
    if (direction[i] == 'L') {
        cur = cur->llink;
    } else {
        cur = cur->rlink;
    }
}
```

```
}  
if (cur == NULL || i != strlen(direction)) {  
    printf("Insertion not possible \n");  
    freeNode(*temp);  
    return root;  
}
```

```
}  
if (cur == NULL) {  
    if (direction[i - 1] == 'L')  
        prev->llink = temp;  
    else  
        prev->rlink = temp;  
}  
return root;
```

```
}  
void preorder(NODE root) {  
    if (root == NULL) {  
        printf("the item is %d \n", root->info);  
        preorder(root->llink);  
        preorder(root->rlink);  
    }
}
```

```
void inorder(NODE root) {
    if (root != NULL) {
        inorder(root->l.link);
        printf("The item is %d in ", root->info);
        inorder(root->r.link);
    }
}
```

```
void postorder(NODE root) {
    if (root != NULL) {
        postorder(root->l.link);
        postorder(root->r.link);
        printf("The item is %d in ", root->info);
    }
}
```

```
void display(NODE root, int i) {
    int j;
    if (root != NULL) {
        display(root->l.link, i+1);
        for (j=1, j <= i; j++)
            printf("   ");
        printf("%d \n", root->info);
        display(root->r.link, i+1);
    }
}
```

```
int main() {
    NODE root = NULL;
    int choice, i, item;
    for(;;) {
        printf("1. Insert In 2. Preorder In 3. Inorder In 4. Postorder In 5.
Display \n");
        printf("Enter the choice \n");
        scanf("%d \n", &choice);
        switch(choice) {

```

```
case 1: printf("Enter the item : \n");
        scanf("%d", &item);
        root = insert(item, root);
        break;
```

```
case 2: if (root == NULL) {  
    printf("Tree is empty \n");  
}  
else {  
    printf("Given tree is... ");  
    display(root, 1);  
    printf("The preorder traversal is: \n");  
    preorder(root);  
}  
break;
```

```
case 3: if (root == NULL) {  
    printf("Tree is empty \n");  
}  
else {  
    printf("Given tree is.. ");  
    display(root, 1);  
    printf("The inorder traversal is \n");  
    inorder(root);  
}  
break;
```

```
case 4: if (root == NULL) {  
    printf("Tree is empty ");  
}  
else {  
    printf("Given tree is.. ");  
    display(root, 1);  
    printf("The postorder traversal is \n");  
    postorder(root);  
}  
break;
```

```
case 5: display(root, 1);  
break;
```

```
default: printf("Invalid choice entered \n");  
exit(0);
```

classmate

Date _____
Page _____

88

return 0;

f

Binary_search_tree - [tree_binaryserach.dev] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
binary_search_tree_lp10.cpp
```

```
1 #include <stdio.h>
2 #include <conio.h>
3 #include <malloc.h>
4 #include <process.h>
5 struct node
6 {
7     int info;
8     struct node *rlink;
9     struct node *llink;
10 };
11 typedef struct node *NODE;
12 NODE getnode()
13 {
14     NODE x;
15     x=(NODE) malloc(sizeof(struct node));
16     if(x==NULL)
17     {
18         printf("mem full\n");
19         exit(0);
20     }
21     return x;
22 }
23 void freenode(NODE x)
24 {
25     free(x);
26 }
27 NODE insert(NODE root, int item)
28 {
29     NODE temp, cur, prev;
30     temp=getnode();
31     temp->rlink=NULL;
32     temp->llink=NULL;
33     temp->info=item;
34     if(root==NULL)
35         return temp;
36     prev=NULL;
37     cur=root;
38     while(cur!=NULL)
```

Compiler Resources Compile Log Debug Find Results

Line: 158 Col: 26 Sel: 0 Lines: 167 Length: 2640 Insert Done parsing in 0.032 seconds

Type here to search

14:35 21-12-2020

Binary_search_tree - [tree_binaryserach.dev] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
binary_search_tree_lp10.cpp
```

```
37 | cur=root;
38 | while(cur!=NULL)
39 | {
40 |     prev=cur;
41 |     cur=(item<cur->info)?cur->l link:cur->r link;
42 |
43 |     if(item<prev->info)
44 |         prev->l link=temp;
45 |     else
46 |         prev->r link=temp;
47 |     return root;
48 |
49 void display(NODE root, int i)
50 {
51     int j;
52     if(root!=NULL)
53     {
54         display(root->r link, i+1);
55         for(j=0; j<i; j++)
56             printf("    ");
57         printf("%d\n", root->info);
58         display(root->l link, i+1);
59     }
60 }
61 NODE delete_that(NODE root, int item)
62 {
63     NODE cur, parent, q, suc;
64     if(root==NULL)
65     {
66         printf("empty\n");
67         return root;
68     }
69     parent=NULL;
70     cur=root;
71     while(cur!=NULL&&item!=cur->info)
72     {
73         parent=cur;
74         cur=(item<cur->info)?cur->l link:cur->r link;
```

Compiler Resources Compile Log Debug Find Results

Line: 158 Col: 26 Sel: 0 Lines: 167 Length: 2640 Insert Done parsing in 0.032 seconds

Type here to search

14:35 21-12-2020

Binary_search_tree - [tree_binaryserach.dev] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
binary_search_tree_lp10.cpp
```

```
73 | parent=cur;
74 | cur=(item<cur->info)?cur->llink:cur->rlink;
75 |
76 | if (cur==NULL)
77 | {
78 |     printf("not found\n");
79 |     return root;
80 | }
81 | if (cur->llink==NULL)
82 |     q=cur->rlink;
83 | else if (cur->rlink==NULL)
84 |     q=cur->llink;
85 | else
86 | {
87 |     suc=cur->rlink;
88 |     while(suc->llink!=NULL)
89 |         suc=suc->llink;
90 |     suc->llink=cur->llink;
91 |     q=cur->rlink;
92 | }
93 | if (parent==NULL)
94 |     return q;
95 | if (cur==parent->llink)
96 |     parent->llink=q;
97 | else
98 |     parent->rlink=q;
99 | freenode(cur);
100 | return root;
101 |
102 void preorder(NODE root)
103 {
104 {
105 if (root!=NULL)
106 {
107     printf("%d\n",root->info);
108     preorder(root->llink);
109     preorder(root->rlink);
110 }
```

Compiler Resources Compile Log Debug Find Results

Line: 158 Col: 26 Sel: 0 Lines: 167 Length: 2640 Insert Done parsing in 0.032 seconds

Type here to search

14:36 21-12-2020

Binary_search_tree - [tree_binaryserach.dev] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
binary_search_tree_lp10.cpp
```

```
109     pre_order( root->rlink );
110 }
111 }
112 void post_order( NODE root )
113 {
114     if( root !=NULL )
115     {
116         post_order( root->llink );
117         post_order( root->rlink );
118         printf( "%d\n", root->info );
119     }
120 }
121 void in_order( NODE root )
122 {
123     if( root !=NULL )
124     {
125         in_order( root->llink );
126         printf( "%d\n", root->info );
127         in_order( root->rlink );
128     }
129 }
130 }
131 }
132 int main()
133 {
134     int item_choice;
135     NODE root=NULL;
136     system("cls");
137     for(;;)
138     {
139         printf( "\n1.insert\n2.display\n3.pre_order\n4.post_order\n5.in_order \n6.delete\n7.exit\n" );
140         printf( "enter the choice\n" );
141         scanf( "%d", &choice );
142         switch( choice )
143         {
144             case 1: printf( "enter the item\n" );
145             scanf( "%d", &item );
146             root = insert( root, item );
147         }
148     }
149 }
```

Compiler Resources Compile Log Debug Find Results

Line: 158 Col: 26 Sel: 0 Lines: 167 Length: 2640 Insert Done parsing in 0.032 seconds

Type here to search

14:36 21-12-2020

Binary_search_tree - [tree_binaryserach.dev] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

```
binary_search_tree_lp10.cpp
```

```
131 }
132 int main()
133 {
134     int item_choice;
135     NODE root=NULL;
136     system("cls");
137     for(;;)
138     {
139         printf("\n1.insert\n2.display\n3.pre_order\n4.post_order\n5.in_order\n6.delete\n7.exit\n");
140         printf("enter the choice\n");
141         scanf("%d", &choice);
142         switch(choice)
143         {
144             case 1: printf("enter the item\n");
145             scanf("%d", &item);
146             root=insert(root, item);
147             break;
148             case 2: display(root, 0);
149             break;
150             case 3: preorder(root);
151             break;
152             case 4: postorder(root);
153             break;
154             case 5: inorder(root);
155             break;
156             case 6: printf("enter the item\n");
157             scanf("%d", &item);
158             root=delete_hat(root, item);
159             break;
160             default: exit(0);
161             break;
162         }
163     }
164     return 0;
165 }
```

Compiler Resources Compile Log Debug Find Results

Line: 158 Col: 26 Sel: 0 Lines: 167 Length: 2640 Insert Done parsing in 0.032 seconds

Type here to search

14:36 21-12-2020

C:\Users\sohan\Desktop\C Programs\Data Structures Lab\tree binary search\tree_binaryserach.exe

```
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
1
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
2
Give direction to insert..
1
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
3
Give direction to insert..
r
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
4
Give direction to insert..
ll
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
```



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\tree binary search\tree_binaryserach.exe

```
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
5
Give direction to insert..
lr
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
6
Give direction to insert..
rl
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
1
Enter the item:
7
Give direction to insert..
rr
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
5
      7
     3   6
    1   5
   2   4
1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display
```



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\tree binary search\tree_binaryserach.exe

2.Preorder
3.Inorder
4.Postorder
5.Display
Enter the choice:
2

Given tree is.. 7
 3
 6
 1
 5
 2
 4

The preorder traversal is:

the item is 1
the item is 2
the item is 4
the item is 5
the item is 3
the item is 3
the item is 6
the item is 7

1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display

Enter the choice:

3
Given tree is.. 7
 3
 6
 1
 5
 2
 4

The inorder traversal is:

The item is 4
The item is 2
The item is 5
The item is 1
The item is 6
The item is 3
The item is 7

1.Insert
2.Preorder
3.Inorder
4.Postorder
5.Display

Enter the choice:

4

Type here to search



C:\Users\sohan\Desktop\C Programs\Data Structures Lab\tree binary search\tree_binaryserach.exe

3.Inorder

4.Postorder

5.Display

Enter the choice:

4

Given tree is... 7

 3

 6

 1

 5

 2

 4

The postorder traversal is

The item is4

The item is5

The item is2

The item is6

The item is7

The item is3

The item is1

1.Insert

2.Preorder

3.Inorder

4.Postorder

5.Display

Enter the choice:

5

 7

 3

 6

 1

 5

 2

 4

1.Insert

2.Preorder

3.Inorder

4.Postorder

5.Display

Enter the choice:

6

Invalid choice entered.

Process exited after 226.4 seconds with return value 0

Press any key to continue . . .



Type here to search



15:00
ENG
21-12-2020

