## **Experiment No.1**

**Aim: -**

### Write a programs to show the implementation of insertion in array

**Source Code: -**

1 #include "stdio.h"

2 #include "conio.h"

3

4 **void** main()

5 {

6 **int** k,i=0,n;

7 printf("Enter the Size of array = ");

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

9 **int** A[n];

10 printf("Enter the Your Elements = \n");

11 **for**(i;i<n;i++)

12 {

13 scanf("%d",&A[i]);

14 }

15 printf("You Entered\n");

16 **for**(i=0;i<n;i++)

17 {

18 printf("%d\t",A[i]);

19 }

20 printf("\nenter the place where New element is inserted = ");

21 scanf("%d",&k);

22 **int** a=n+0;

23 **for**(**int** j=n;j>=k-1;j--)

24 {

25 A[j+1]=A[j];

26

27 }

28

29 printf("Enter The new Element = ");

30 scanf("%d",&A[k-1]);

31 printf("\nNew Array\n");

32 **for**(i=0;i<n+1;i++)

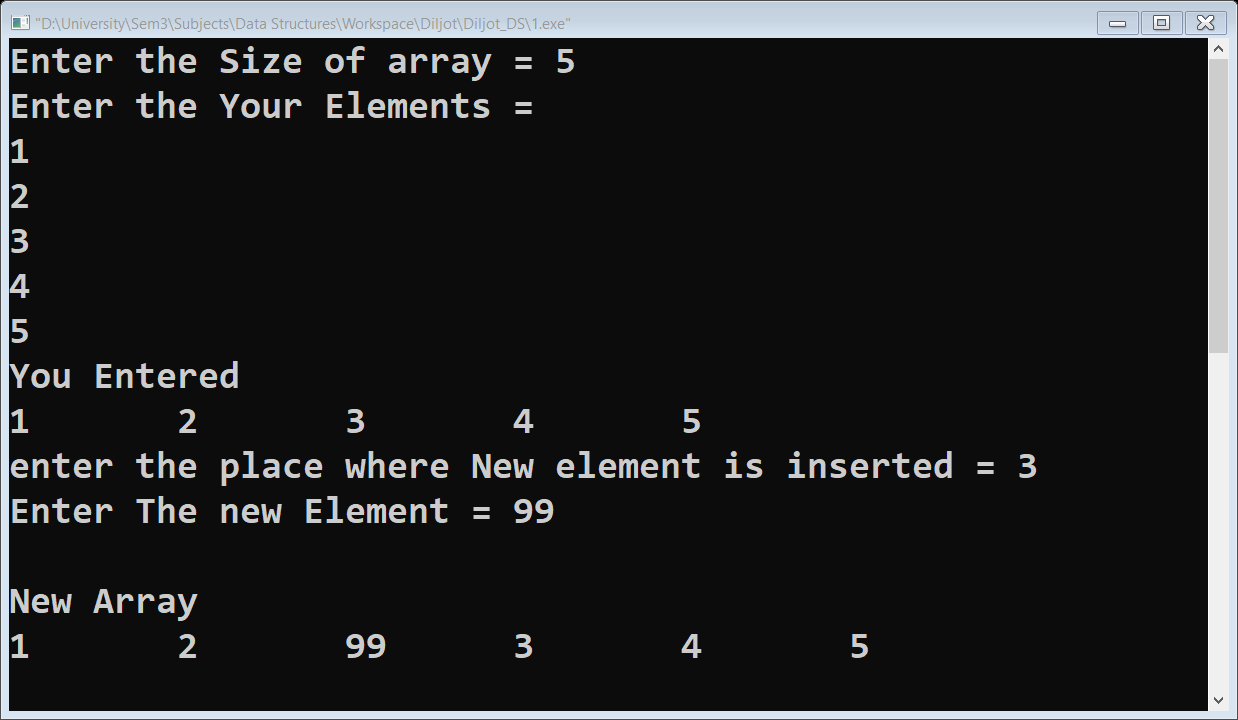
33 {

34 printf("%d\t",A[i]);

35 }

36 getch();

37 }

**Output: -**

## **Experiment No.2**

**Aim: -**

### Write a program to show the implementation of deletion in array

**Source Code: -**

1 #include "stdio.h"

2 #include "conio.h"

3

4 **void** main()

5 {

6 **int** k,i=0,n;

7 printf("Enter the Number Elements = ");

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

9 **int** A[n];

10 printf("Enter the Your Elements = \n");

11 **for**(i;i<n;i++)

12 {

13 scanf("%d",&A[i]);

14 }

15 printf("You Entered \n");

16 **for**(i=0;i<n;i++)

17 {

18 printf("%d\t",A[i]);

19 }

20 printf("\nEnter the place of deletion = ");

21 scanf("%d",&k);

22 **for**(**int** j=k-1;j<n;j++)

23 {

24 A[j]=A[j+1];

25 }

26 printf("New Array \n");

27 **for**(i=0;i<n-1;i++)

28 {

29 printf("%d\t",A[i]);

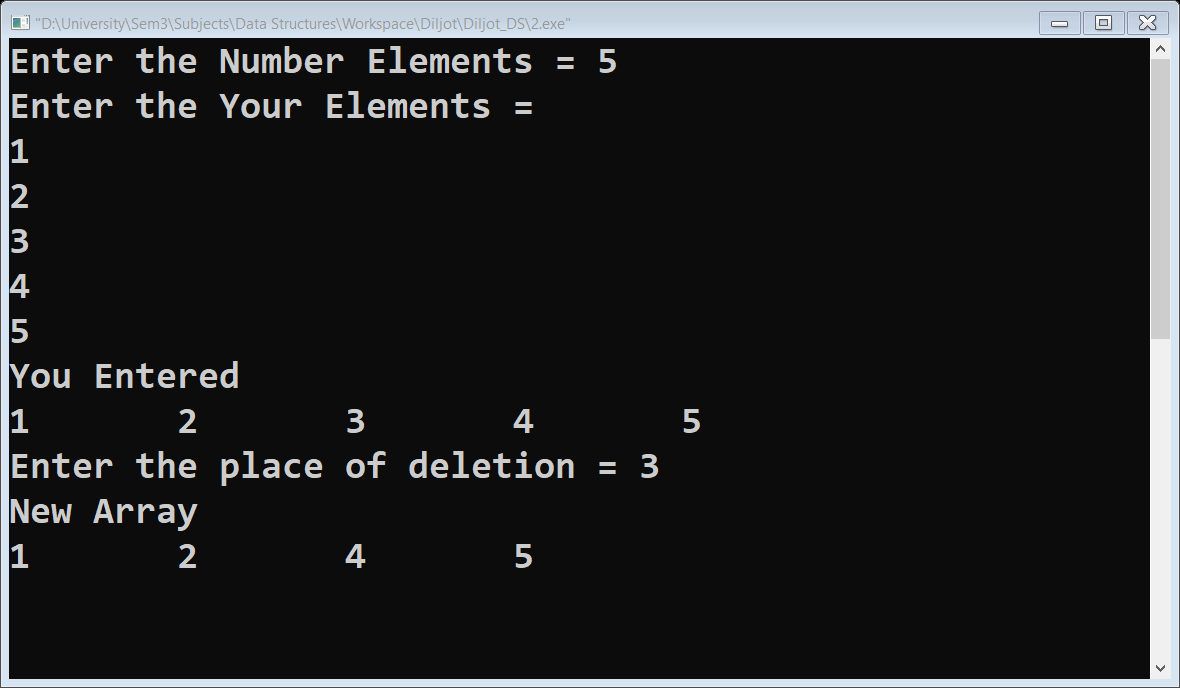
30 }

31 getch();

32

33 }

**Output: -**



## **Experiment No.3**

**Aim: -**

### Write a program to show the implementation of linear search in array

**Source Code: -**

1 #include<stdio.h>

2 #include<conio.h>

3

4 **void** main()

5 {

6 **int** size,Elements,Item,Status=0,ArrayLocation,MemoryLocation;

7 printf("Enter the size of array = ");

8 scanf("%d",&size);

9 **int** array[size];

10 printf("Enter the no. of Elements = ");

11 scanf("%d",&Elements);

12 **for**(**int** i=0;i<Elements;i++)

13 {

14 scanf("%d",&array[i]);

15

16 }

17 printf("Your Entered ");

18 **for**(**int** i=0;i<Elements;i++)

19 {

20 printf("\t%d",array[i]);

21 }

22 printf("\nEnter item you want to search = ");

23 scanf("%d",&Item);

24 **for**(**int** i=0;i<Elements;i++)

25 {

26 **if**(Item==array[i]){

27 Status=1;

28 ArrayLocation=i+1;

29 MemoryLocation&array[i];

30 }

31 }

32 **if** (Status==1){

33 printf("\nItem Found");

34 printf("\nItem Location in array %d",ArrayLocation);

35 printf("\nItem Location in Memory %d",MemoryLocation);

36 }

37

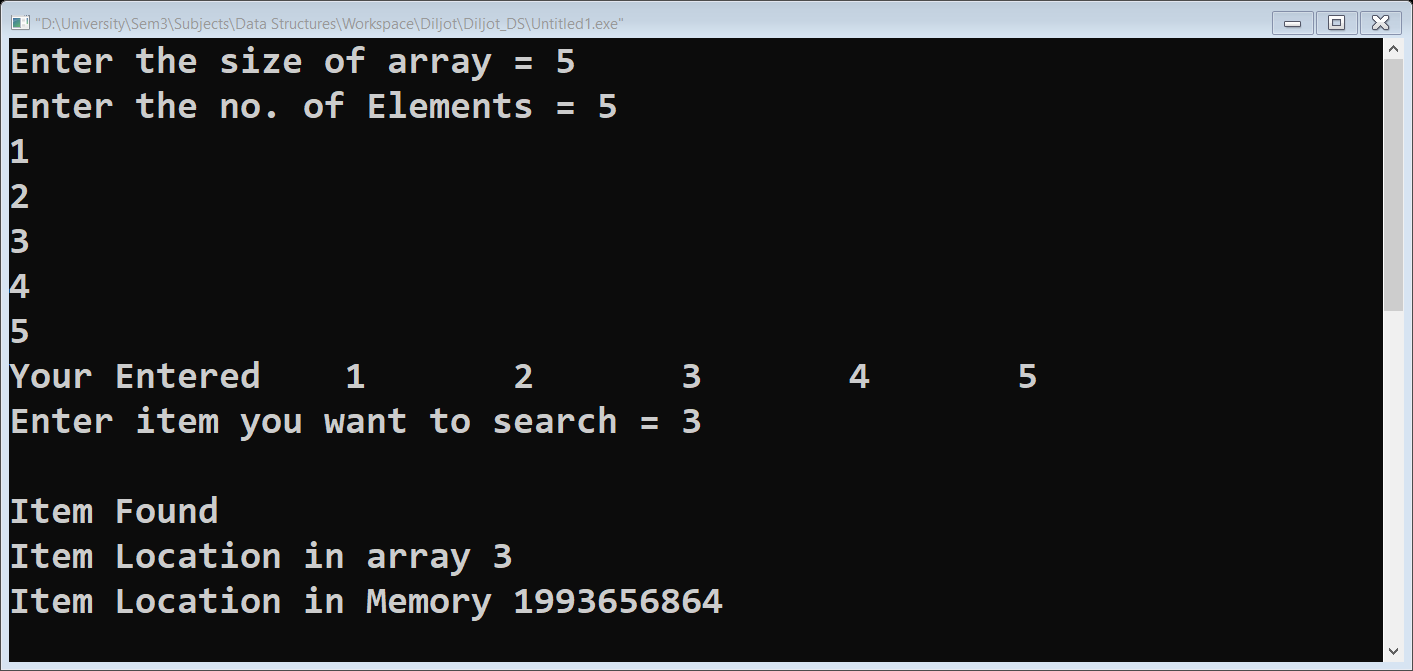
38 **else**{

39 printf("\nItem Not found");

40 }

41 getch();

42 }

**Output: -**

## **Experiment No.4**

**Aim: -**

### Write a program to show the implementation of creation of Linked List

**Source Code: -**

1 /\*Creation of Linked List\*/

2 #include "stdio.h"

3 #include "conio.h"

4 #include "stdlib.h"

5

6 //Linked List is declared here

7 **struct** node

8 {

9 **int** info;

10 **struct** node \*link;

11 };

12 **struct** node \*first;

13

14 **void** main()

15 {

16 **char** ch;

17 **void** create();

18 create();

19 **void** print();

20 printf("\nPress (Y/N) for printing linked list\n");

21 ch =getch();

22 **if**(ch=='Y'||ch=='y')

23 {

24 printf("Your Linked List\n");

25 print();

26 }

27 getch();

28 }

29 /\*

30 create() function helps us to create Linked list(of desired length)

31 and insert data node by node

32 \*/

33 **void** create()

34 {

35 **struct** node \*ptr,\*cpt;

36 **char** ch;

37 ptr =(**struct** node\*) malloc (**sizeof** (**struct** node));

38 printf("Input First Node information = ");

39 scanf("%d",&ptr->info);

40 first=ptr;

41 **do**

42 {

43 cpt=(**struct** node\*) malloc(**sizeof** (**struct** node));

44 printf("\nInput next node information = ");

45 scanf("%d",&cpt->info);

46 ptr->link =cpt;

47 ptr = cpt;

48 printf("Press (Y/N) for more node ");

49 ch =getch();

50 }

51 **while**(ch=='Y'||ch=='y');

52 ptr->link = NULL;

53 }

54 /\*

55 print() function helps us in traversing the linked list

56 and printing the data in it

57 \*/

58 **void** print()

59 {

60 **struct** node \*ptr;

61 printf("\n%d ",first->info);

62 ptr=first->link;

63 **while**(ptr)

64 {

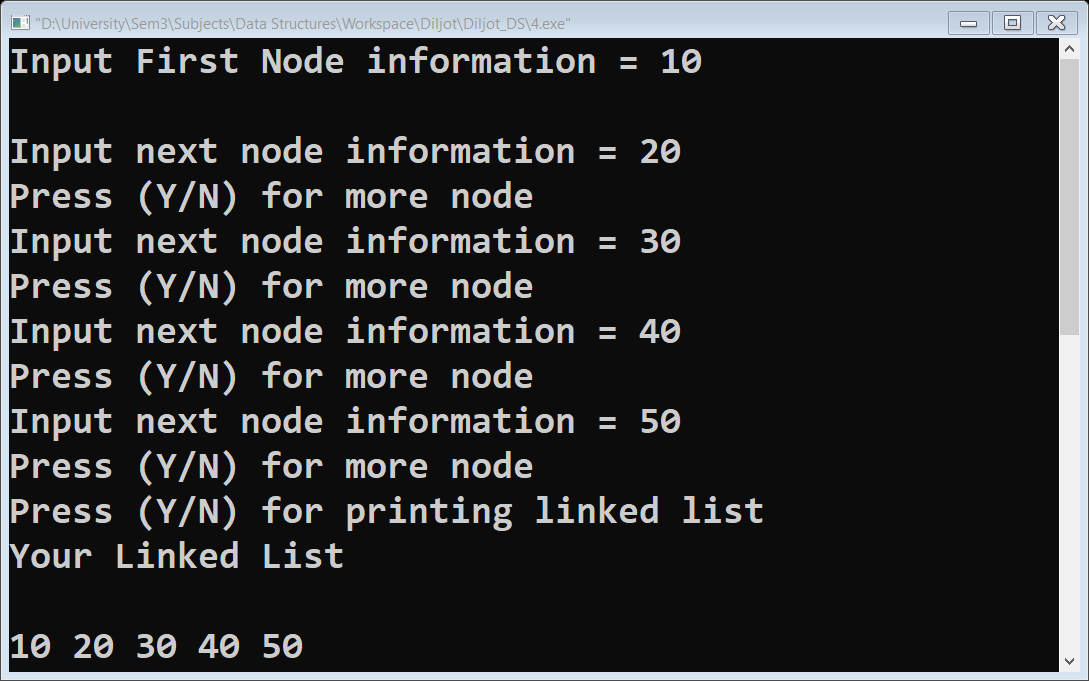
65 printf("%d ",ptr->info);

66 ptr=ptr->link;

67 }

68 }

**Output: -**

****

## **Experiment No.5**

**Aim: -**

### Write a program to show the implementation of Searching in Linked List

**Source Code: -**

1 /\*Searching Linked List\*/

2 #include "stdio.h"

3 #include "conio.h"

4 #include "stdlib.h"

5

6

7 //Linked List is declared here

8 **struct** node

9 {

10 **int** info;

11 **struct** node \*link;

12 };

13 **struct** node \*first;

14

15 **void** main()

16 {

17 **char** ch,chx,chz;

18 **void** create();

19 **void** print();

20 **int** search(**int** x);

21 create();

22 printf("\nPress (Y/N) for printing linked list\n");

23 ch =getch();

24 **if**(ch=='Y'||ch=='y')

25 {

26 printf("Your Linked List\n");

27 print();

28 }

29 printf("\nPress (Y/N) for Searching item in Linked list");

30 chz =getch();

31 **if**(chz=='Y'||chz=='y')

32 {

33 **int** f,k;

34 printf("\nEnter the item You want to search = ");

35 scanf("%d",&k);

36 f=search(k);

37 **if**(f==0)

38 printf("\nItem not found");

39 }

40 printf("\nPress (Y/N) if you want to restart\n");

41 chx=getch();

42 **if**(chx=='y'||chx=='Y')

43 {

44 system("cls");

45 main();

46 }

47 getch();

48 }

49 /\*

50 create() function helps us to create Linked list(of desired length)

51 and insert data node by node

52 \*/

53 **void** create()

54 {

55 **struct** node \*ptr,\*cpt;

56 **char** ch;

57 ptr =(**struct** node\*) malloc (**sizeof** (**struct** node));

58 printf("Input First Node information = ");

59 scanf("%d",&ptr->info);

60 first=ptr;

61 **do**

62 {

63 cpt=(**struct** node\*) malloc(**sizeof** (**struct** node));

64 printf("\nInput next node information = ");

65 scanf("%d",&cpt->info);

66 ptr->link =cpt;

67 ptr = cpt;

68 printf("Press (Y/N) for more node ");

69 ch =getch();

70 }

71 **while**(ch=='Y'||ch=='y');

72 ptr->link = NULL;

73 }

74 /\*

75 print() function helps us in traversing the linked list

76 and printing the data in it

77 \*/

78 **void** print()

79 {

80 **struct** node \*ptr;

81 printf("\n%d ",first->info);

82 ptr=first->link;

83 **while**(ptr)

84 {

85 printf("%d ",ptr->info);

86 ptr=ptr->link;

87 }

88

89 }

90 /\*

91 search() function helps to search the item in linked list

92 \*/

93 **int** search(**int** x)

94 {

95 **int** f=0;

96 **struct** node \*ptr;

97 ptr=first->link;

98 **while**(ptr)

99 {

100 **if**(x==ptr->info)

101 {

102 f=1;

103 printf("\nItem Found");

104 printf("\nItem Found at Memory Address = %d",&ptr->link);

105 }

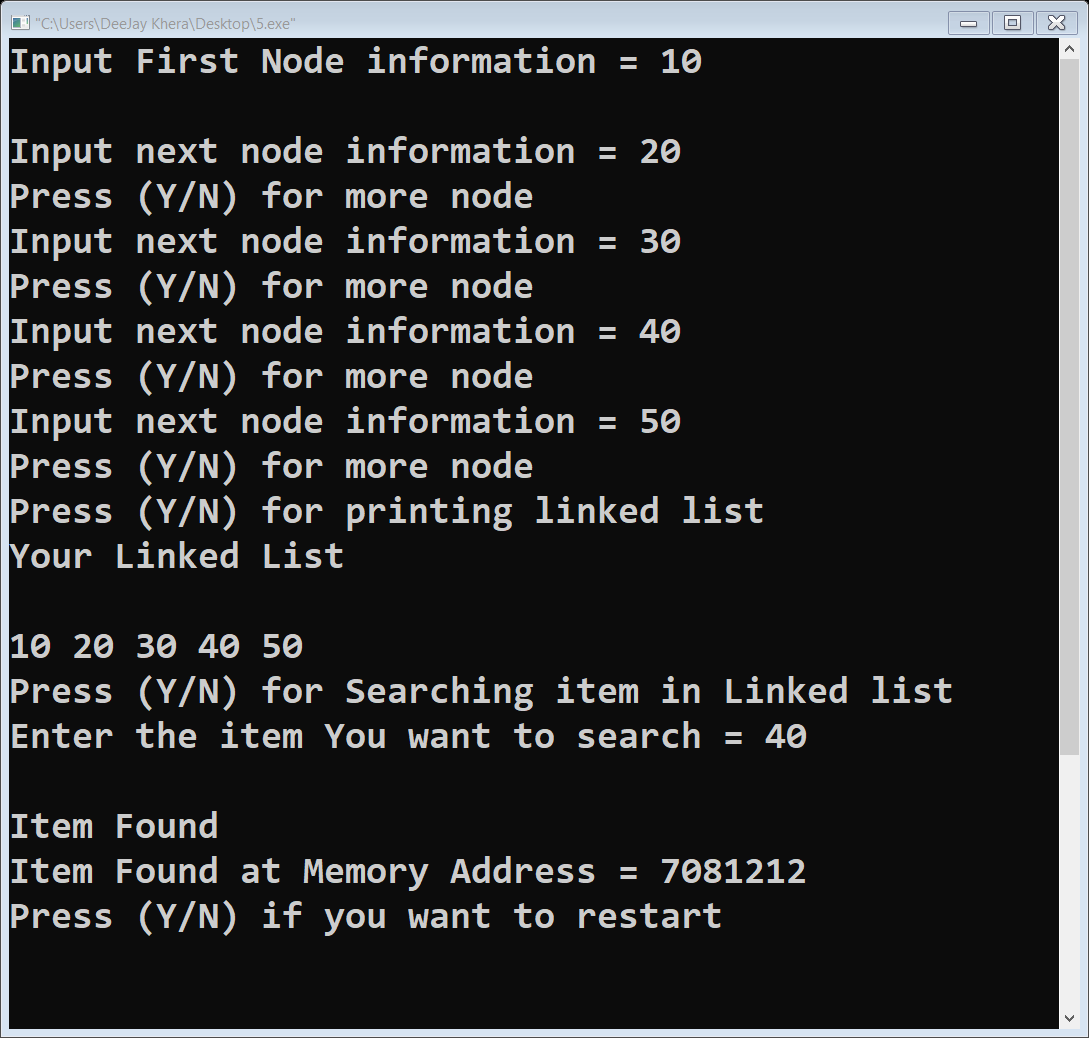
106 ptr=ptr->link;

107 }

108 **return** f;

109 }

**Output: -**

****

## **Experiment No.6**

**Aim: -**

### Write a program to show the implementation of Deletion of node in Linked List

**Source Code: -**

1 /\*Deletion Linked List\*/

2 #include "stdio.h"

3 #include "conio.h"

4 #include "stdlib.h"

5

6 //Linked List is declared here

7 **struct** node

8 {

9 **int** info;

10 **struct** node \*link;

11 };

12 **struct** node \*start;

13

14 **void** main()

15 {

16 **char** ch,chx;

17 **void** create();

18 **void** print();

19 **void** deletion(**int** n);

20 **int** search(**int** x);

21 **int** n;

22 create();

23 printf("\nPress (Y/N) for printing linked list\n");

24 ch =getch();

25 **if**(ch=='Y'||ch=='y')

26 {

27 printf("Your Linked List\n");

28 print();

29 }

30 printf("\nEnter the Location of Deletion = ");

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

32 deletion(n);

33 printf("\nPress (Y/N) for printing linked list\n");

34 ch =getch();

35 **if**(ch=='Y'||ch=='y')

36 {

37 printf("Your Linked List\n");

38 print();

39 }

40 printf("\nPress (Y/N) if you want to restart\n");

41 chx=getch();

42 **if**(chx=='y'||chx=='Y')

43 {

44 system("cls");

45 main();

46 }

47 getch();

48 }

49 /\*

50 create() function helps us to create Linked list(of desired length)

51 and insert data node by node

52 \*/

53 **void** create()

54 {

55 **struct** node \*ptr,\*cpt;

56 **char** ch;

57 ptr =(**struct** node\*) malloc (**sizeof** (**struct** node));

58 printf("Input First Node information = ");

59 scanf("%d",&ptr->info);

60 start=ptr;

61 **do**

62 {

63 cpt=(**struct** node\*) malloc(**sizeof** (**struct** node));

64 printf("\nInput next node information = ");

65 scanf("%d",&cpt->info);

66 ptr->link =cpt;

67 ptr = cpt;

68 printf("Press (Y/N) for more node ");

69 ch =getch();

70 }

71 **while**(ch=='Y'||ch=='y');

72 ptr->link = NULL;

73 }

74 /\*

75 print() function helps us in traversing the linked list

76 and printing the data in it

77 \*/

78 **void** print()

79 {

80 **struct** node \*ptr;

81 printf("\n%d ",start->info);

82 ptr=start->link;

83 **while**(ptr)

84 {

85 printf("%d ",ptr->info);

86 ptr=ptr->link;

87 }

88

89 }

90 **void** deletion(**int** n)

91 {

92 **struct** node \*ptr,\*cpt;

93 **int** i=2;

94 **if** (n==1)

95 {

96 start=start->link;

97 }

98 **else if** (n !=1)

99 {

100 cpt=start;

101 ptr=start->link;

102 **do**

103 {

104 **if**(i==n)

105 {

106 cpt->link=ptr->link;

107 }

108 cpt=cpt->link;

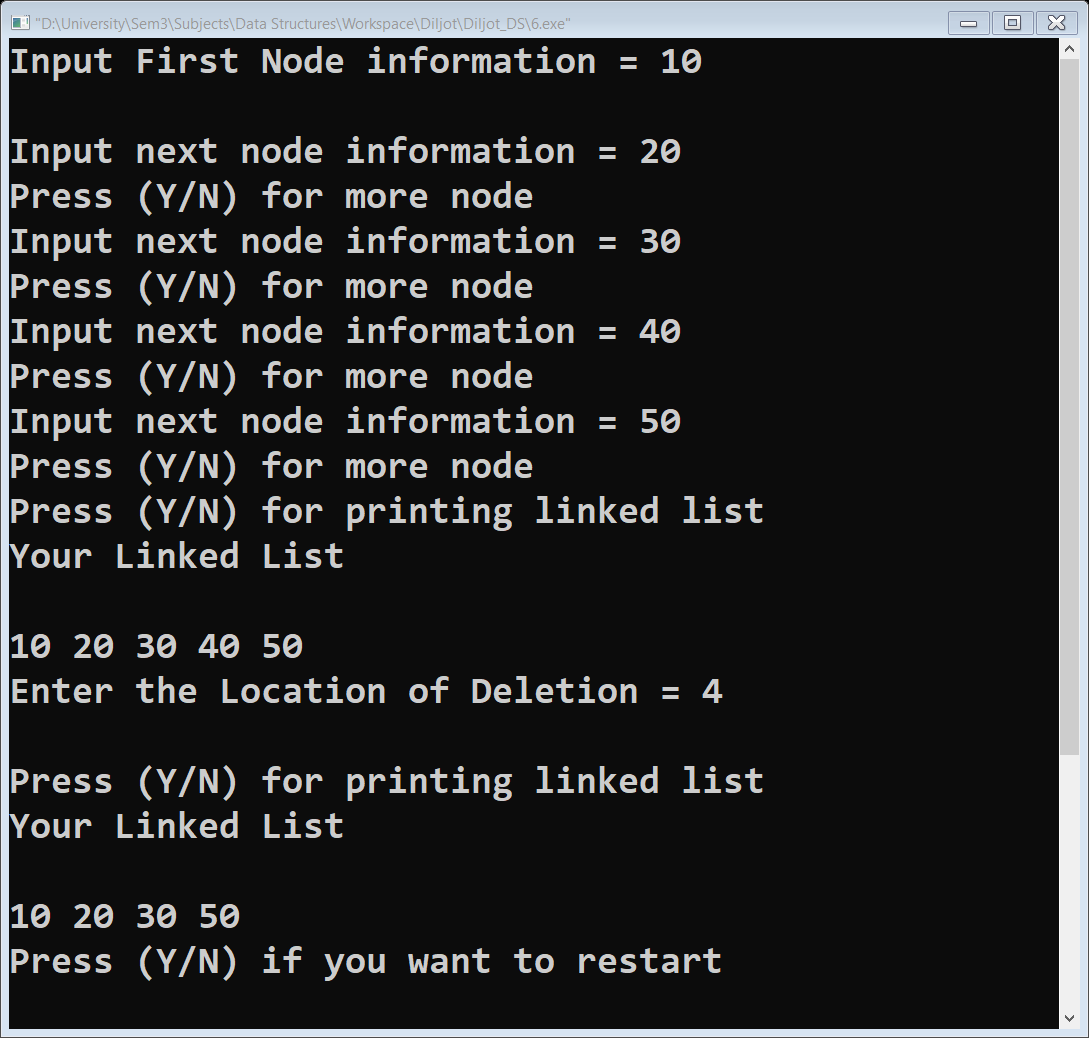
109 ptr=ptr->link;

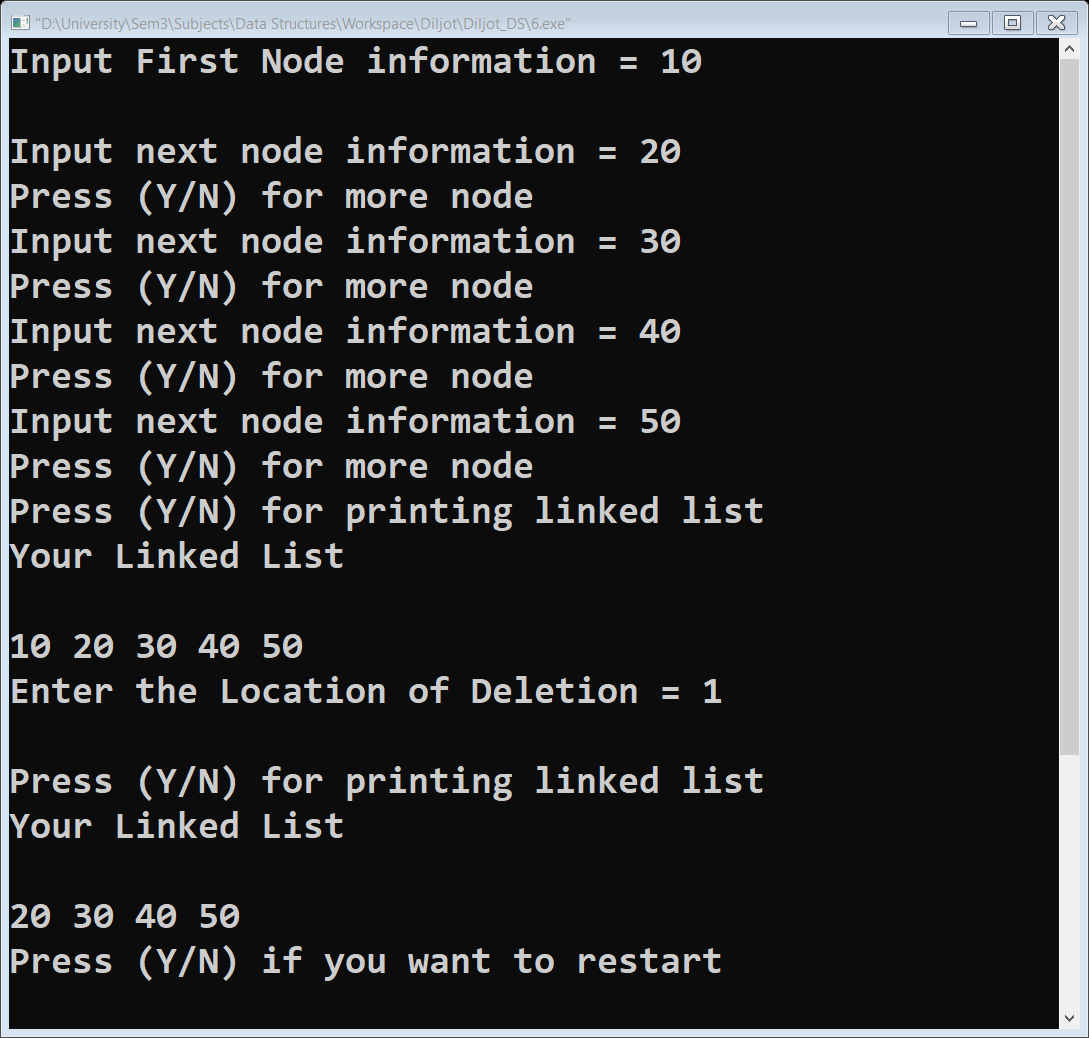
110 i++;

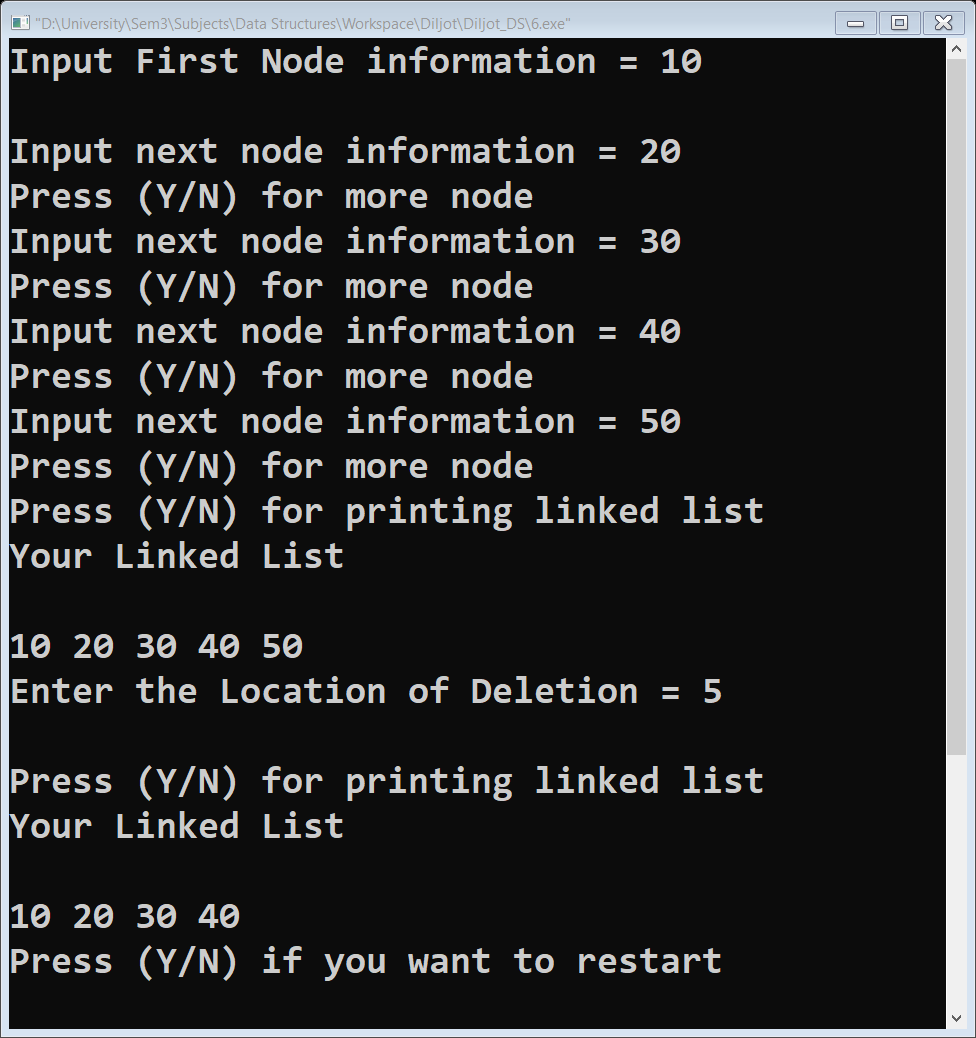
111 }**while**(ptr);

112 }

113 }

**Output: -**





1. Deletion of last node
2. Deletion of First node
3. Deletion after Given Node

## **Experiment No. 7**

**Aim: -**

### Write a program to show the implementation of insertion of node in Linked List

**Source Code: -**

1 /\*Insertion Linked List\*/

2 #include "stdio.h"

3 #include "conio.h"

4 #include "stdlib.h"

5

6 //Linked List is declared here

7 **struct** node

8 {

9 **int** info;

10 **struct** node \*link;

11 };

12 **struct** node \*first;

13

14 **void** main()

15 {

16 **int** n;

17 **char** ch;

18 **void** create();

19 create();

20 **void** print();

21 **void** insertion(**int** n);

22 printf("\nPress (Y/N) for printing linked list\n");

23 ch =getch();

24 **if**(ch=='Y'||ch=='y')

25 {

26 printf("Your Linked List\n");

27 print();

28 ch='n';

29 }

30 printf("\nEnter The Location =");

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

32 insertion(n);

33 printf("\nPress (Y/N) for printing linked list\n");

34 ch =getch();

35 **if**(ch=='Y'||ch=='y')

36 {

37 printf("Your Linked List\n");

38 print();

39 }

40 getch();

41 }

42 /\*

43 create() function helps us to create Linked list

44 (of desired length) and insert data node by node

45 \*/

46 **void** create()

47 {

48 **struct** node \*ptr,\*cpt;

49 **char** ch;

50 ptr =(**struct** node\*) malloc (**sizeof** (**struct** node));

51 printf("Input First Node information = ");

52 scanf("%d",&ptr->info);

53 first=ptr;

54 **do**

55 {

56 cpt=(**struct** node\*) malloc(**sizeof** (**struct** node));

57 printf("\nInput next node information = ");

58 scanf("%d",&cpt->info);

59 ptr->link =cpt;

60 ptr = cpt;

61 printf("Press (Y/N) for more node ");

62 ch =getch();

63 }

64 **while**(ch=='Y'||ch=='y');

65 ptr->link = NULL;

66 }

67 /\*

68 print() function helps us in traversing the

69 linked list and printing the data in it

70 \*/

71 **void** print()

72 {

73 **struct** node \*ptr;

74 printf("\n%d ",first->info);

75 ptr=first->link;

76 **while**(ptr)

77 {

78 printf("%d ",ptr->info);

79 ptr=ptr->link;

80 }

81 }

82

83 **void** insertion(**int** n)

84 {

85 **struct** node \*ptr,\*cpt,\*NEW;

86 ptr=first->link;87 **if**(n==1)//insert as First Node

88 {

89 NEW=(**struct** node\*) malloc(**sizeof** (**struct** node));

90 printf("\nInput next node information = ");

91 scanf("%d",&NEW->info);

92 NEW->link=first;

93 first=NEW;

94 }

95 **else**

96 { **int** i=2;

97 cpt=first;

98 ptr=first->link;

99 **do**

100 {

101 **if**(n==i)//Insert after Given Node

102 {

103 NEW=(**struct** node\*) malloc(**sizeof** (**struct** node));

104 printf("\nInput next node information = ");

105 scanf("%d",&NEW->info);

106 NEW->link=ptr;

107 cpt->link=NEW;

108 }

109 cpt=cpt->link;

110 ptr=ptr->link;

111 i++;

112 }**while**(ptr);

113 **if**(n==i)//Insert as Last Node

114 {

115 NEW=(**struct** node\*) malloc(**sizeof** (**struct** node));

116 printf("\nInput next node information = ");

117 scanf("%d",&NEW->info);

118 cpt->link=NEW;

119 NEW->link=NULL;

120 }

121 **else if**(n>i)

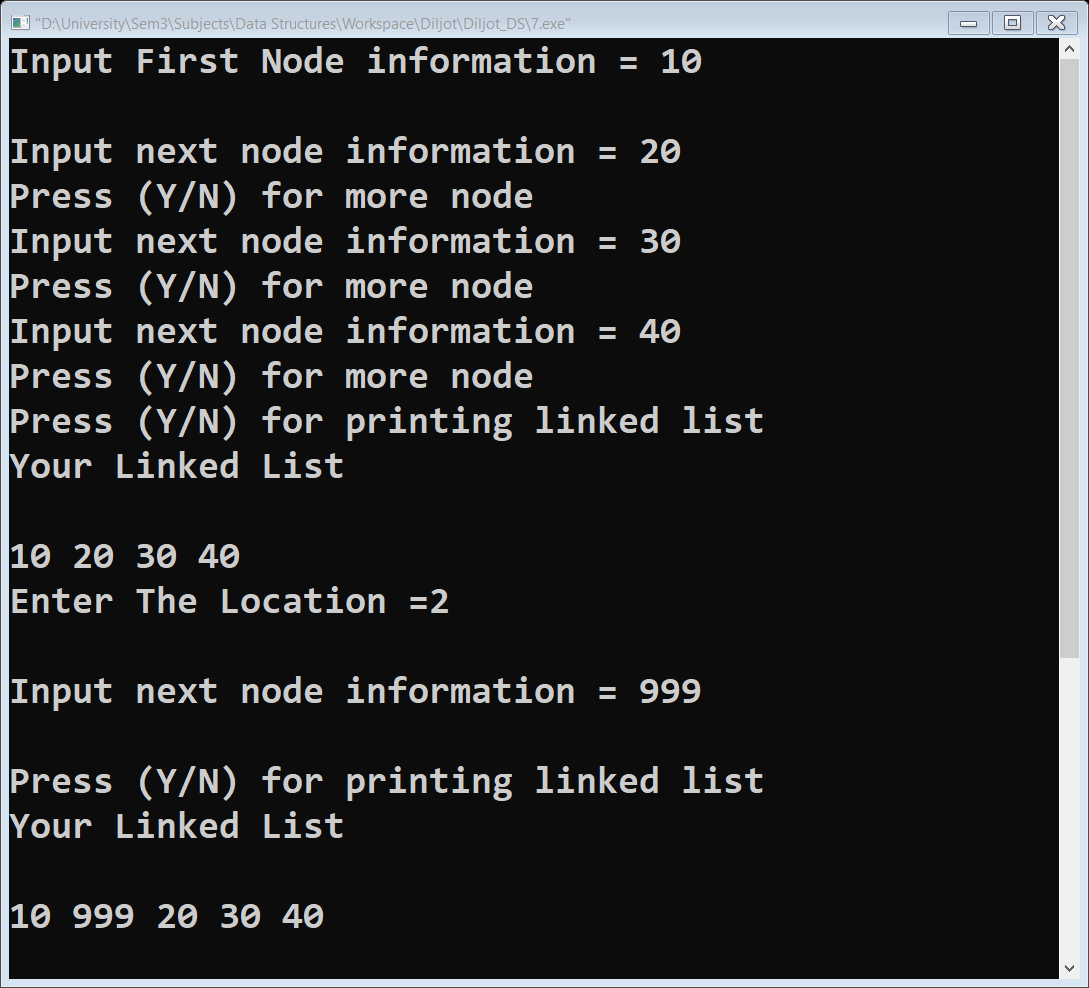
122 {

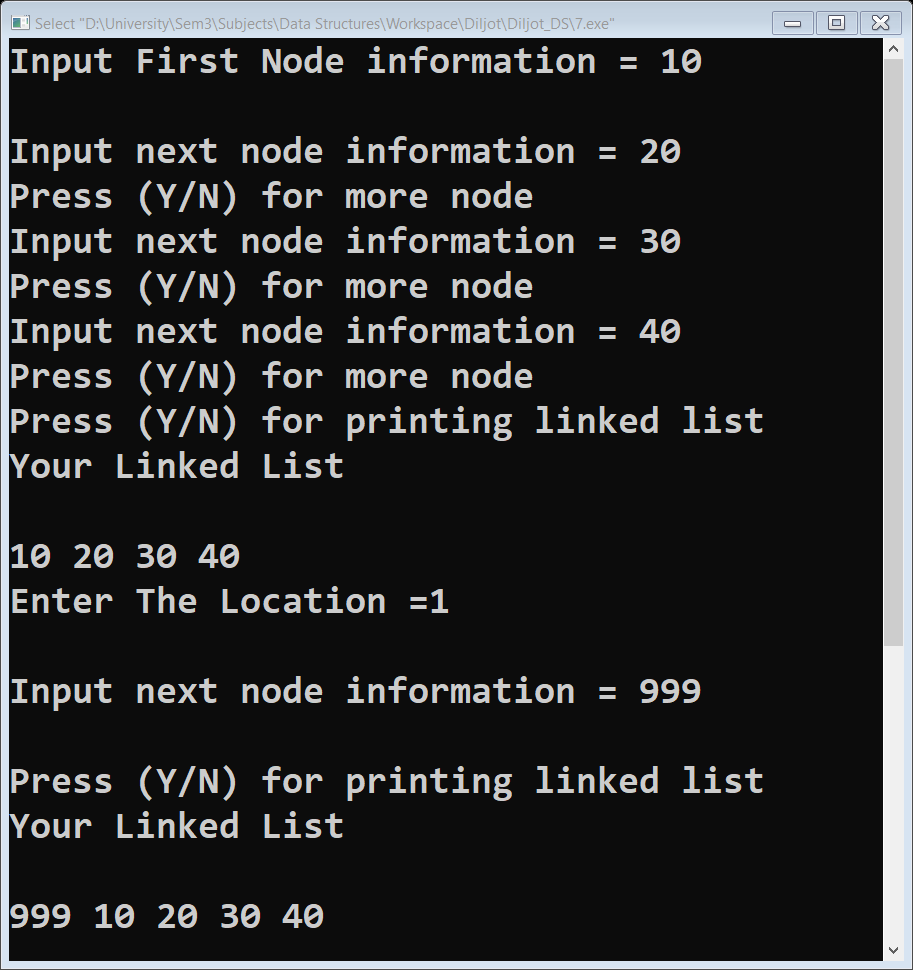
123 printf("\nEnter Location carefully");

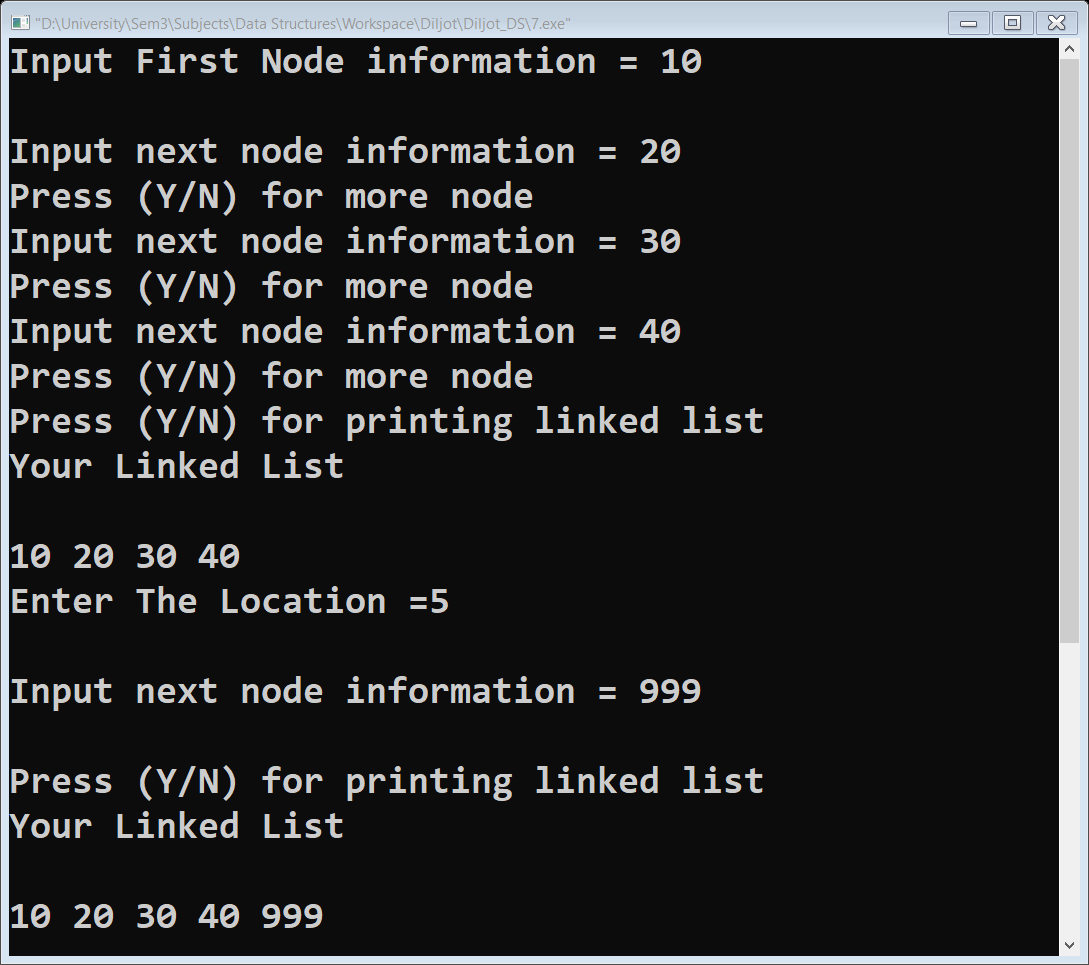
124 }

125 }

126 }

**Output: -**





1. Insertion at last node

(b) Insertion as First node

1. Insertion of node in middle

**Experiment No. 8**

**Aim: -**

Write a program to show the implementation of stack push pop Operation

**Source Code: -**

1 /\*

2 \*Stack Push and pop

3 \*/

4 #include "stdio.h"

5 #include "conio.h"

6 #define MAX 5

7 **int** top=-1 ,stack[MAX];

8 **void** push()

9 {

10 **int** val;

11 **if**(top==MAX-1)

12 {

13 printf("\nstack is full");

14 }

15 **else**

16 {

17 printf("\nEnter Element to push");

18 scanf("%d",&val);

19 top=top+1;

20 stack[top]=val;

21 }

22 }

23 **int** pop()

24 {

25 **if**(top==-1)

26 {

27 printf("\nStack is empty");

28 }

29 **else**

30 {

31 printf("\nPoped Element is %d",stack[top]);

32 **return** stack[top];

33 top=top-1;

34 }

35 }

36 **void** display()

37 {

38 **int** i;

39 **if**(top==-1)

40 {

41 printf("\nStack is empty");

42 }

43 **else**

44 {

45 printf("\nStack is .....\n]");

46 **for**(**int** j =1;j>=0;j--)

47 {

48 printf("%d ",stack[j]);

49 }

50 }

51 }

52 **int** main()

53 {

54 **int** ch;

55 printf("\nSTACK PUSH & POP \n1.Push\n2.Pop\n3.Display\n4.Exit\n");

56 **while**(1)

57 {

58 printf("Enter your choice (1-4): " );

59 scanf("%d",&ch);

60 **switch**(ch)

61 {

62 **case** 1:push();

63 **break**;

64 **case** 2:pop();

65 **break**;

66 **case** 3:display();

67 **break**;

68 **case** 4:exit(0);

69 **break**;

70 **default**:printf("Wrong Choice\n");

71 **break**;

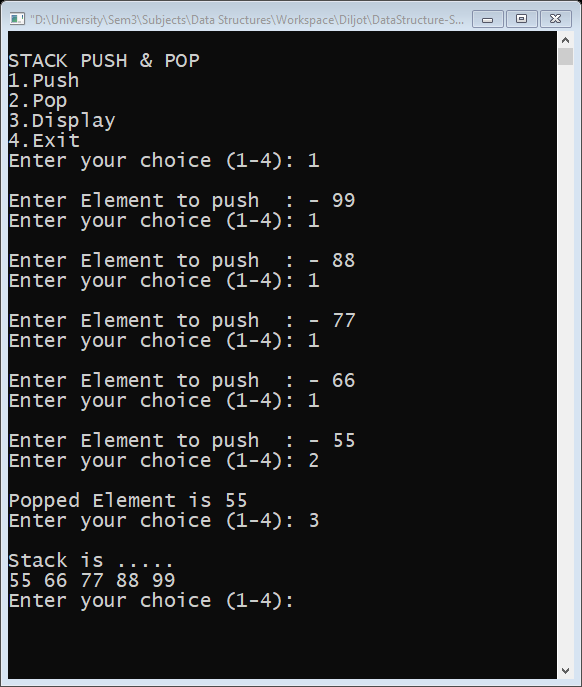
72

73 }

74 }

75 }

**Output: -**

****

**Experiment No. 9**

**Aim: -**

Write a program to show the implementation of stack push pop Operation

**Source Code: -**

1 /\*

2 \*Convert infix to postfix using Stack

3 \*/

4 #include "stdio.h "

5 #include "stdlib.h " /\* for exit() \*/

6 #include " ctype.h " /\* for isdigit(char ) \*/

7 #include "string.h "

8

9 #define SIZE 100

10

11 **char** stack[SIZE];

12 **int** top = -1;

13

14 /\* define push operation \*/

15

16 **void** push(**char** item)

17 {

18 **if** (top >= SIZE - 1)

19 {

20 printf("\nStack Overflow.");

21 }

22 **else**

23 {

24 top = top + 1;

25 stack[top] = item;

26 }

27 }

28

29 /\* define pop operation \*/

30 **char** pop()

31 {

32 **char** item;

33

34 **if** (top < 0)

35 {

36 printf("stack under flow: invalid infix expression");

37 getchar();

38 exit(1);

39 }

40 **else**

41 {

42 item = stack[top];

43 top = top - 1;

44 **return** (item);

45 }

46 }

47

48 **int** is\_operator(**char** symbol)

49 {

50 **if** (symbol == '^' || symbol == '\*' || symbol == '/' || symbol == '+' || symbol == '-')

51 {

52 **return** 1;

53 }

54 **else**

55 {

56 **return** 0;

57 }

58 }

59

60 **int** precedence(**char** symbol)

61 {

62 **if** (symbol == '^') /\* exponent operator, highest precedence\*/

63 {

64 **return** (3);

65 }

66 **else if** (symbol == '\*' || symbol == '/')

67 {

68 **return** (2);

69 }

70 **else if** (symbol == '+' || symbol == '-') /\* lowest precedence \*/

71 {

72 **return** (1);

73 }

74 **else**

75 {

76 **return** (0);

77 }

78 }

79

80 **void** InfixToPostfix(**char** infix\_exp[], **char** postfix\_exp[])

81 {

82 **int** i, j;

83 **char** item;

84 **char** x;

85

86 push('('); /\* push '(' onto stack \*/

87 strcat(infix\_exp, ")"); /\* add ')' to infix expression \*/

88 i = 0;

89 j = 0;

90 item = infix\_exp[i]; /\* initialize before loop\*/

91

92 **while** (item != '\0') //run loop till end of infix expression \* /

93 {

94 **if** (item == '(')

95 {

96 push(item);

97 }

98 **else if** (isdigit(item) || isalpha(item))

99 {

100 postfix\_exp[j] = item; /\*add operand symbol to postfix expr \*/

101 j++;

102 }

103 **else if** (is\_operator(item) == 1) /\* means symbol is operator \*/

104 {

105 x = pop();

106 **while** (is\_operator(x) == 1 && precedence(x) >= precedence(item))

107 {

108 postfix\_exp[j] = x; /\* so pop all higher precendence operator and \*/

109 j++;

110 x = pop(); /\* add them to postfix expresion \*/

111 }

112 push(x);

113 push(item); /\* push current oprerator symbol onto stack \*/

114 }

115 **else if** (item == ')') /\* if current symbol is ')' then \*/

116 {

117 x = pop(); /\* pop and keep popping until \*/

118 **while** (x != '(') /\* '(' encounterd \*/

119 {

120 postfix\_exp[j] = x;

121 j++;

122 x = pop();

123 }

124 }

125 **else**

126 {

127 /\* if current symbol is neither operand not '(' nor ')' and nor operator \*/

128 printf("\nInvalid infix Expression.\n");

129 getchar();

130 exit(1);

131 }

132 i++;

133 item = infix\_exp[i]; /\* go to next symbol of infix expression \*/

134 }

135

136 **if** (top > 0)

137 {

138 printf("\nInvalid infix Expression.\n");

139 getchar();

140 exit(1);

141 }

142 postfix\_exp[j] = '\0';

143

144 }

145

146 /\* main function begins \*/

147 **int** main()

148 {

149 **char** infix[SIZE], postfix[SIZE];

150

151 printf("ASSUMPTION: The infix expression contains single letter \nvariables and single digit constants only.\n");

152 printf("\nEnter Infix expression : ");

153 gets(infix);

154

155 InfixToPostfix(infix, postfix);

156 printf("Postfix Expression: ");

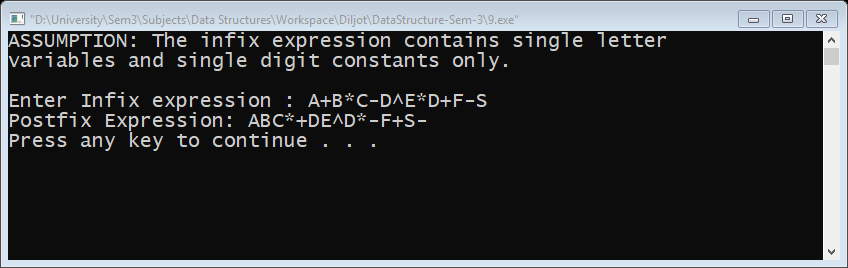
157 puts(postfix); /\* print postfix expression \*/

158 system("pause");

159 **return** 0;

160 }

**Output: -**

****