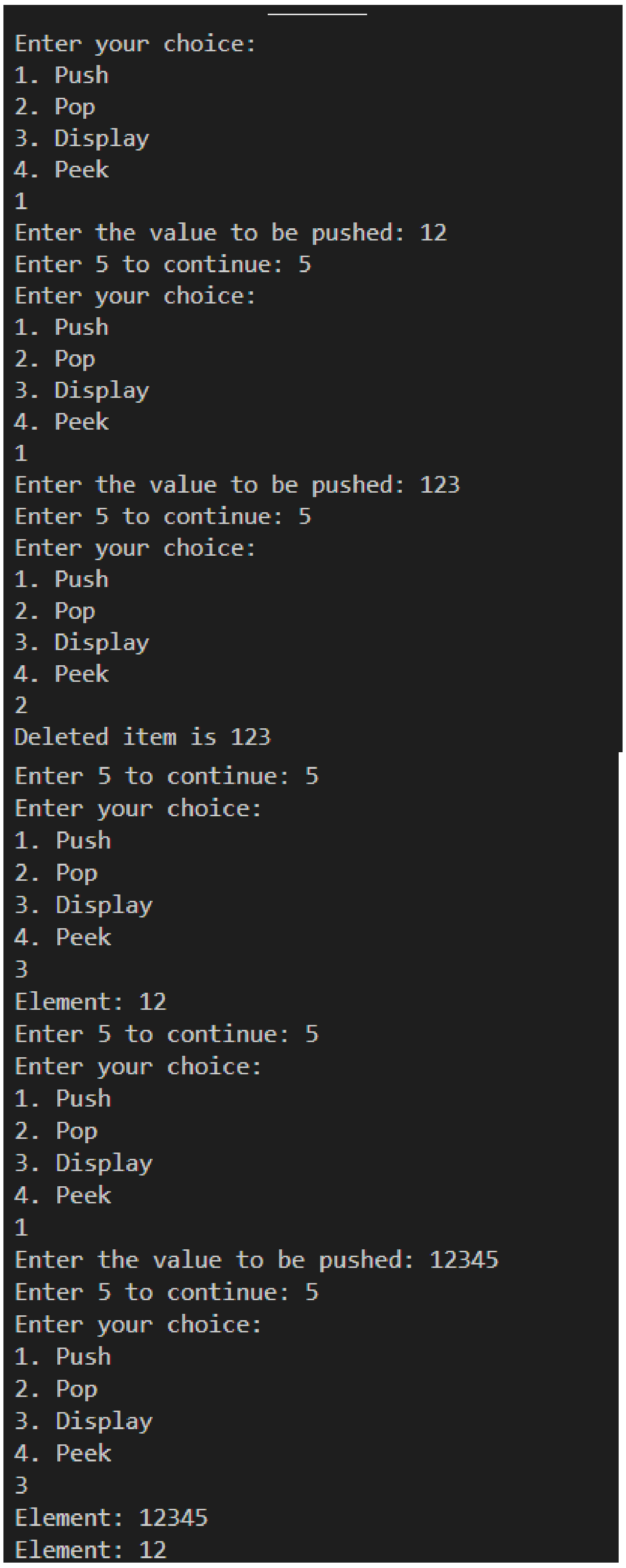
Experiment 1: Implementation of stack using menu driven approach.

Code:
```
#include<stdio.h>
int arr[10];
int top = -1;
int n = 10;

void push(int val) {
   if(top == n - 1) {
      printf("Stack is Full\n");
   } else {
      top = top + 1;
      arr[top] = val;
   }}

void peek() {
   if(top == -1)
      printf("\nStack is empty\n");
    else
      printf("\nTop element is %d\n", arr[top]);
  }

void pop() {
   int temp;
   if(top == -1)
      printf("Stack is empty\n");
   else {
      temp = arr[top];
      top = top - 1;
      printf("Deleted item is %d\n", temp);}}

void display() {

   if(top == -1) {
      printf("Stack is empty\n");    }
 else {
      for(int i = top; i >= 0; i--) {
         printf("Element: %d\n", arr[i]);       }}}
```

```
int main() {
    int choice, e, val;
    do {
        printf("Enter your choice:\n1. Push\n2. Pop\n3. Display\n4. Peek\n");
        scanf("%d", &choice);
        switch(choice) {
            case 1:
                printf("Enter the value to be pushed: ");
                scanf("%d", &val);
                push(val);
                break;

            case 2:
                pop();
                break;

            case 3:
                display();
                break;

            case 4:
                peek();
                break;

            default:
                printf("Invalid choice\n");
        }

        printf("Enter 5 to continue: ");
        scanf("%d", &e);
    } while(e == 5);
    return 0;
}
```

Output:



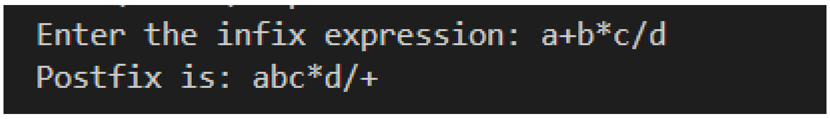
```
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
1
Enter the value to be pushed: 12
Enter 5 to continue: 5
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
1
Enter the value to be pushed: 123
Enter 5 to continue: 5
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
2
Deleted item is 123
Enter 5 to continue: 5
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
3
Element: 12
Enter 5 to continue: 5
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
1
Enter the value to be pushed: 12345
Enter 5 to continue: 5
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
3
Element: 12345
Element: 12
```

Experiment 2: Implementation of Infix to Postfix conversion

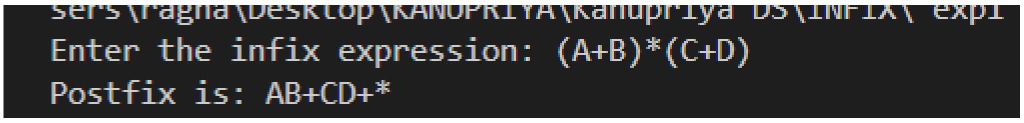
Code:

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

#define MAX 30
char stack[MAX];
int top = -1;

void push(char val) {
  if (top < MAX - 1) {
    stack[++top] = val;
  }}

char pop() {
  if (top == -1)
    return -1;
  else
    return stack[top--];}

int priority(char ch) {
  switch (ch) {
    case '^':
      return 3;
    case '*':
    case '/':
            return 2;
        case '+':
        case '-':
            return 1;
        default:
            return 0;
    }}
void infix_to_postfix(char infix[], char postfix[]) {
    char temp, x;
    int i = 0, j = 0;
    while (infix[i] != '\0') {
        temp = infix[i];
        if (isalnum(temp)) {
            postfix[j++] = temp;
        } else if (temp == '(') {
            push(temp);
        } else if (temp == ')') {
            while ((x = pop()) != '(') {
                postfix[j++] = x;
            }
        } else {
            while (top != -1 && priority(stack[top]) > priority(temp) ||
                   (priority(stack[top]) == priority(temp))) {
                x = pop();
                postfix[j++] = x;
            }
            push(temp);
        }
        i++; }

    while (top != -1) {
        postfix[j++] = pop();
    }
    postfix[j] = '\0';
}

int main() {
    char infix[MAX];
    char postfix[MAX];
    printf("Enter the infix expression: ");
    scanf("%s", infix);
    infix_to_postfix(infix, postfix);
    printf("Postfix is: %s\n", postfix);
    return 0;
}
```

Output:



```
Enter the infix expression: a+b*c/d
Postfix is: abc*d/+
```



```
sers\ragha\Desktop\KANUPRIYA\Kanupriya DS\INFIX\ expl
Enter the infix expression: (A+B)*(C+D)
Postfix is: AB+CD+*
```

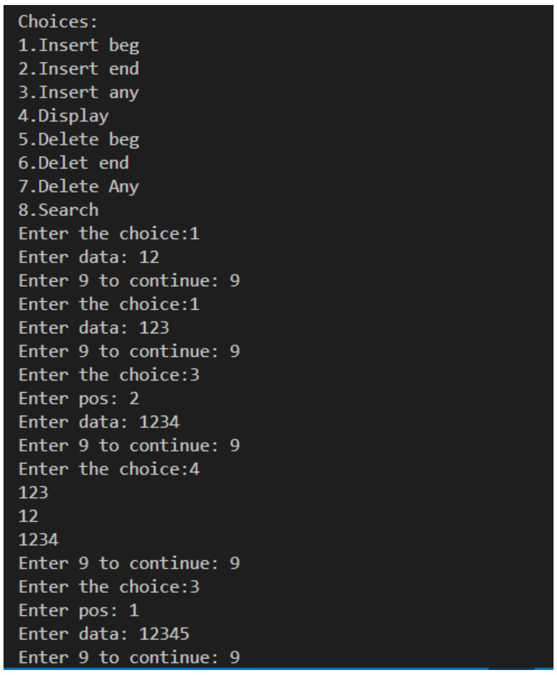
Experiment 3: Implementation of Linked List using menu driven approach.

Code:

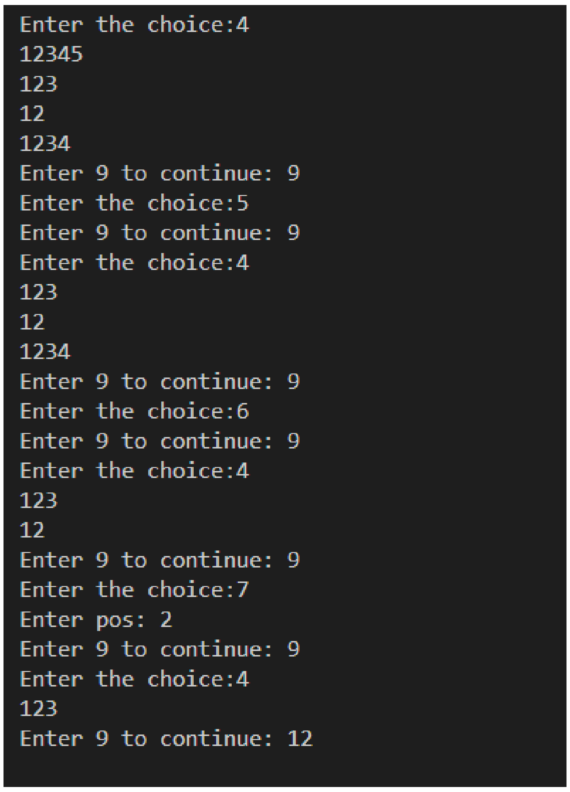
```
#include <stdio.h>
#include <stdlib.h>
typedef struct node {
   int data;
   struct node* next;
} node;
node* head = NULL;
node* temp = NULL;node* newnode = NULL;

void insertBeg(){
   node* newnode = (node*)malloc(sizeof(node));
   if (newnode == NULL) {
      printf("Memory allocation failed\n");
      return;
   }
   printf("Enter data: ");
   scanf("%d", &newnode->data);
   newnode->next = head;
   head = newnode;
}

void inserEnd(){
   node* newnode = (node*)malloc(sizeof(node));
   if (newnode == NULL) {
        printf("Memory allocation failed\n");
        return;
    }
    printf("Enter data: ");
    scanf("%d", &newnode->data);
    newnode->next = 0;
    if (head == NULL) {
        head = newnode;
    } else {
        temp = head;
        while(temp->next != 0) {
            temp = temp->next;
        }
        temp->next = newnode;
    }
}
    void insertAny(){
    int i=1,pos;
    printf("Enter pos: ");
    scanf("%d", &pos);
    if (pos == 1) {
        node* newnode = (node*)malloc(sizeof(node));
        if (newnode == NULL) {
            printf("Memory allocation failed\n");
            return;
        }
        printf("Enter data: ");
        scanf("%d", &newnode->data);
        newnode->next = head;
        head = newnode;
    } else {
        node* newnode = (node*)malloc(sizeof(node));
        if (newnode == NULL) {
            printf("Memory allocation failed\n");
            return;
        }
        printf("Enter data: ");
        scanf("%d", &newnode->data);
        temp = head;
        while (i < pos ) {
            if(temp->next == NULL) {
                printf("Position out of range\n");
                return;
            }
            temp = temp->next;
            i++;
        }
        newnode->next = temp->next;
        temp->next = newnode;
    }
}
    void deleteBeg(){
```

```
    temp=head;

    head=head->next;

    free(temp);

}

    void deleteEnd(){node*prev;

    temp=head;

    while(temp->next!=0){

        prev=temp;

        temp=temp->next;    }

        if(temp==head){head==0;}

        else{prev->next=0;}

        free(temp);

}


    void deleteAny(){

     int i=1,pos;node* nextnode;

    printf("Enter pos: ");

    scanf("%d", &pos);

    temp=head;

    while(i<pos-1){

        temp=temp->next;

        i++;

    }

    if (temp == NULL || temp->next == NULL) {

        printf("Invalid position. Position exceeds list length.\n");

        return;
    }

    nextnode=temp->next;
    temp->next=nextnode->next;
    free(nextnode);
}

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

    void search(){ int i=1,key;node* temp;
    printf("Enter data to be searched: ");
    scanf("%d", &key);
    temp=head;
    if(temp==NULL)
    {
        printf("\nThe list is empty");
    }
    int found=0;
    while((temp!=NULL)&&(found==0))
    {
        if(temp->data!=key)
        temp=temp->next;
        else
        found=1;  }
    if(found==1)
    printf("\nThe element is present\n");
    else
    printf("\nThe element is not present\n"); }
    int main(){
    int choice,e;
    printf("Choices: \n1.Insert beg\n2.Insert end\n3.Insert
any\n4.Display\n5.Delete beg\n6.Delet end\n7.Delete Any\n8.Search\n");
    do {
        printf("Enter the choice:");
        scanf("%d", &choice);
        switch (choice) {
            case 1: insertBeg(); break;
            case 2: inserEnd(); break;
            case 3: insertAny(); break;
            case 4: display(); break;
            case 5: deleteBeg(); break;
            case 6: deleteEnd();break;
            case 7: deleteAny();break;
            case 8: search();break; }
        printf("Enter 9 to continue: ");
        scanf("%d", &e);
    } while (e == 9);
    return 0;
}
```

Output:



```
Choices:
1.Insert beg
2.Insert end
3.Insert any
4.Display
5.Delete beg
6.Delet end
7.Delete Any
8.Search
Enter the choice:1
Enter data: 12
Enter 9 to continue: 9
Enter the choice:1
Enter data: 123
Enter 9 to continue: 9
Enter the choice:3
Enter pos: 2
Enter data: 1234
Enter 9 to continue: 9
Enter the choice:4
123
12
1234
Enter 9 to continue: 9
Enter the choice:3
Enter pos: 1
Enter data: 12345
Enter 9 to continue: 9
```



```
Enter the choice:4
12345
123
12
1234
Enter 9 to continue: 9
Enter the choice:5
Enter 9 to continue: 9
Enter the choice:4
123
12
1234
Enter 9 to continue: 9
Enter the choice:6
Enter 9 to continue: 9
Enter the choice:4
123
12
Enter 9 to continue: 9
Enter the choice:7
Enter pos: 2
Enter 9 to continue: 9
Enter the choice:4
123
Enter 9 to continue: 12
```

Experiment 4: Implementation of stack and queue using Linked List.

Code: QUEUE

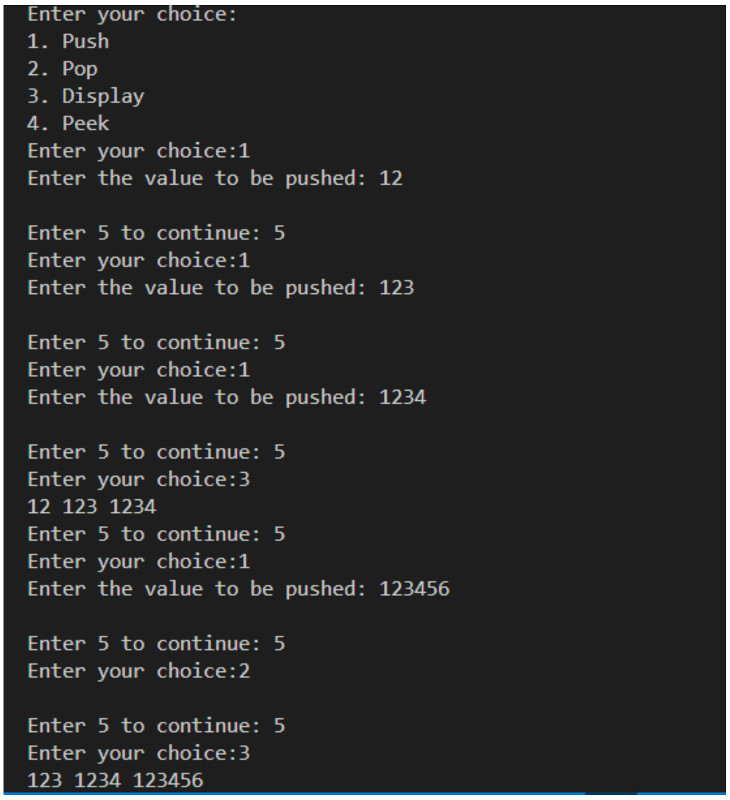
```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
   int data;
   struct node*next;
   }node;
node * front=0;node*rear=0;
void enqueue(int val){node *newnode;
newnode=(node *)malloc(sizeof (node));
newnode->data=val;
newnode->next=0;
   if(front==0&&rear==0){
      front=rear=newnode;
}else{
rear->next=newnode;
rear=newnode;
}
}

void dequeue(){node * temp;
   temp==front;
if(front==0&&rear==0){
      printf("empty");
```

```
}else{
 front=  front->next;
   free(temp);
}
}

void display(){
   node*temp;
   temp=front;
   if(front==0)printf("empty");
   else{
      while(temp!=0){
         printf("%d ",temp->data);
         temp=temp->next;
      }}}
void peek(){
   if(front==0)printf("empty");
   else
   printf(" top is %d\n",front->data);
}
int main() {
   int choice, e, val;
 printf("Enter your choice:\n1. Push\n2. Pop\n3. Display\n4. Peek\n");
   do {
      printf("Enter your choice:");
      scanf("%d", &choice);
```

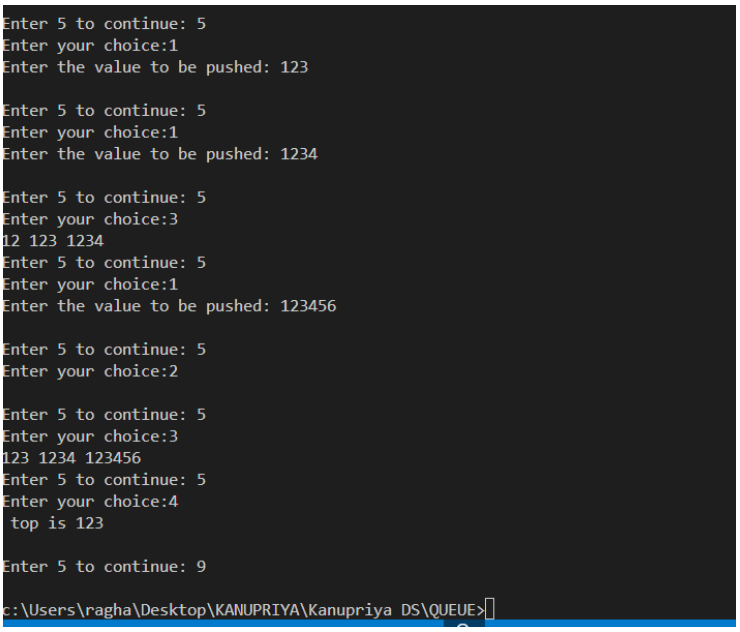
```
        switch(choice) {
            case 1:
                printf("Enter the value to be pushed: ");
                scanf("%d", &val);
                enqueue(val);
                break;

            case 2:
                dequeue();
                break;

            case 3:
                display();
                break;

            case 4:
                peek();
                break;

            default:
                printf("Invalid choice\n");}
        printf("\nEnter 5 to continue: ");
        scanf("%d", &e);
    } while(e == 5);

    return 0;
```

}

Output:



```
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
Enter your choice:1
Enter the value to be pushed: 12

Enter 5 to continue: 5
Enter your choice:1
Enter the value to be pushed: 123

Enter 5 to continue: 5
Enter your choice:1
Enter the value to be pushed: 1234

Enter 5 to continue: 5
Enter your choice:3
12 123 1234
Enter 5 to continue: 5
Enter your choice:1
Enter the value to be pushed: 123456

Enter 5 to continue: 5
Enter your choice:2

Enter 5 to continue: 5
Enter your choice:3
123 1234 123456
```



```
Enter 5 to continue: 5
Enter your choice:1
Enter the value to be pushed: 123

Enter 5 to continue: 5
Enter your choice:1
Enter the value to be pushed: 1234

Enter 5 to continue: 5
Enter your choice:3
12 123 1234
Enter 5 to continue: 5
Enter your choice:1
Enter the value to be pushed: 123456

Enter 5 to continue: 5
Enter your choice:2

Enter 5 to continue: 5
Enter your choice:3
123 1234 123456
Enter 5 to continue: 5
Enter your choice:4
 top is 123

Enter 5 to continue: 9

c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\QUEUE>
```

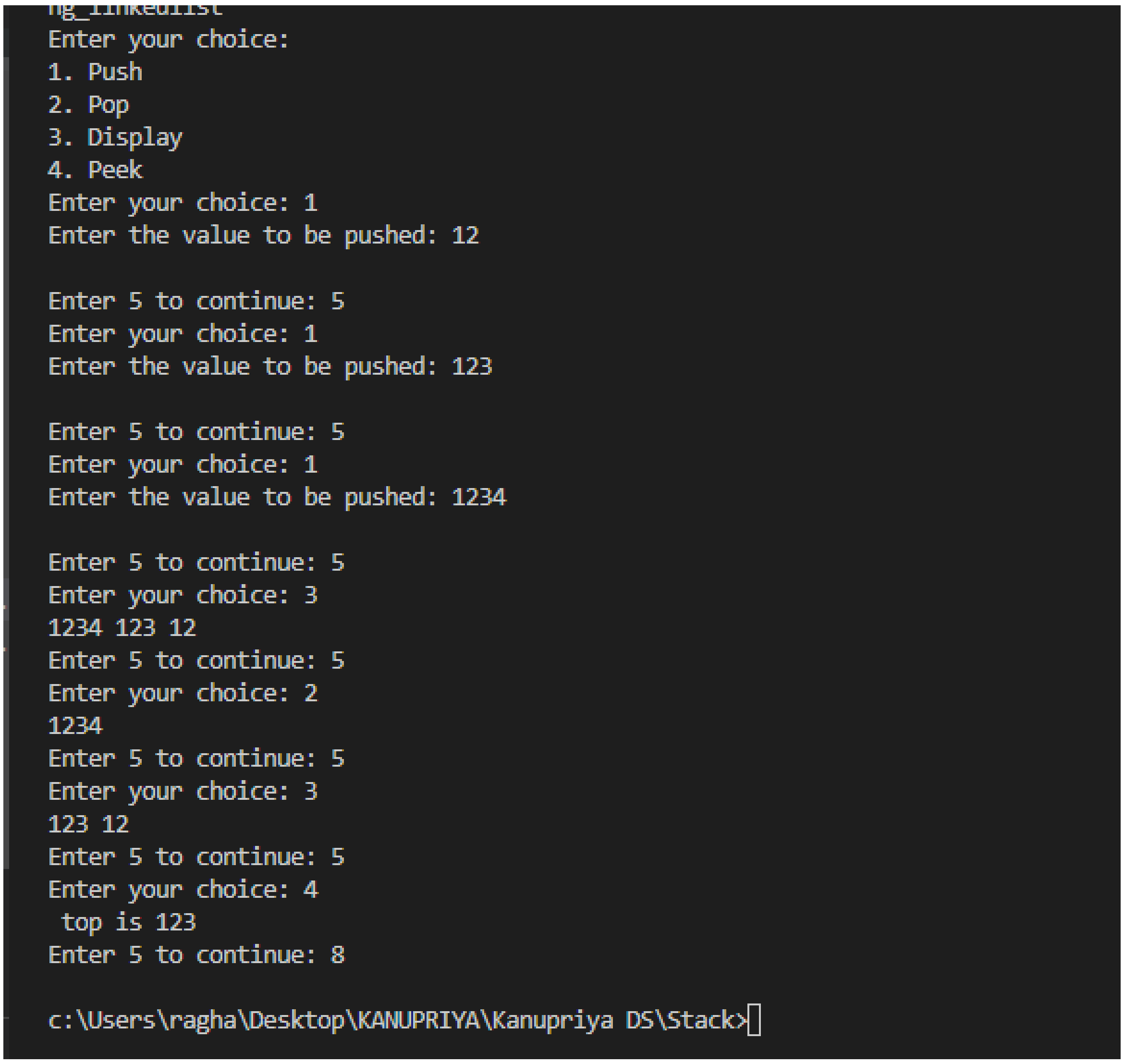
Code: STACK

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

}node;
node*top=0;
void push(int val){
    node*newnode;
    newnode=(node*)malloc(sizeof(node));
    newnode->data=val;
    newnode->next=top;
    top=newnode;
}

void display(){
    node*temp;
    temp=top;
    if(top==0)printf("empty");
    else{
        while(temp!=0){
            printf("%d ",temp->data);
            temp=temp->next;
        }
    }
}

void peek(){
    if(top==0)printf("empty");
    else
    printf(" top is %d",top->data);
}

void pop(){
    node*temp;
    temp=top;
    if(top==0){
        printf("empty");
    }else{
        printf("%d",top->data);
        top=top->next;
        free(temp);
    }}
int main() {
    int choice, e, val;
printf("Enter your choice:\n1. Push\n2. Pop\n3. Display\n4. Peek\n");
    do {
        printf("Enter your choice: ");
```

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

switch(choice) {
    case 1:
        printf("Enter the value to be pushed: ");
        scanf("%d", &val);
        push(val);
        break;

    case 2:
        pop();
        break;

    case 3:
        display();
        break;

    case 4:
        peek();
        break;

    default:
        printf("Invalid choice\n");
}

printf("\nEnter 5 to continue: ");
```

```
        scanf("%d", &e);
    } while(e == 5);


    return 0;
}
```

Output:



```
ng_linkedlist
Enter your choice:
1. Push
2. Pop
3. Display
4. Peek
Enter your choice: 1
Enter the value to be pushed: 12

Enter 5 to continue: 5
Enter your choice: 1
Enter the value to be pushed: 123

Enter 5 to continue: 5
Enter your choice: 1
Enter the value to be pushed: 1234

Enter 5 to continue: 5
Enter your choice: 3
1234 123 12
Enter 5 to continue: 5
Enter your choice: 2
1234
Enter 5 to continue: 5
Enter your choice: 3
123 12
Enter 5 to continue: 5
Enter your choice: 4
 top is 123
Enter 5 to continue: 8

c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\Stack>
```

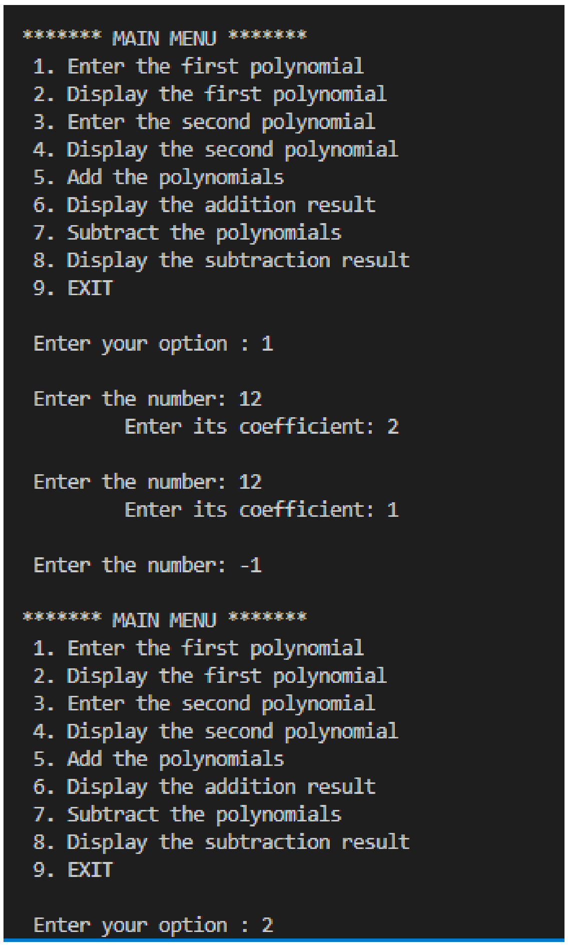
Experiment 5: Implementation of polynomials operations (addition, subtraction) using Linked List.

CODE:

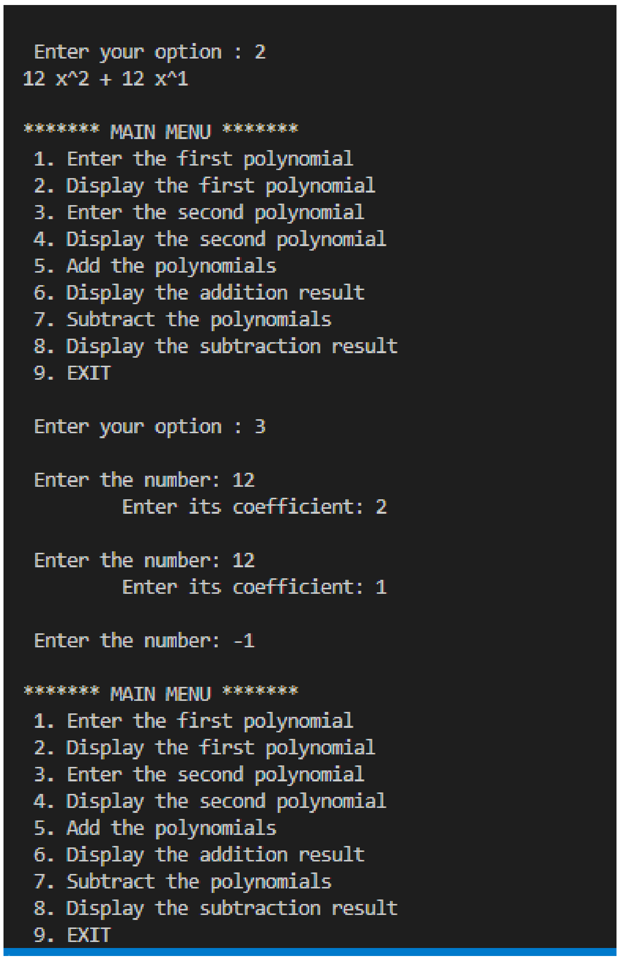
```
#include <stdio.h>

#include <stdlib.h>


struct node {

   int num;

   int coeff;

   struct node* next;

};


struct node* add_node(struct node* start, int n, int c);

struct node* display_poly(struct node* start);

struct node* add_poly(struct node* start1, struct node* start2, struct node*
start3);

struct node* sub_poly(struct node* start1, struct node* start2, struct node*
start4);


int main() {

   struct node *start1 = NULL, *start2 = NULL, *start3 = NULL, *start4 = NULL;

   int option, n, c;


   do {

      printf("\n******* MAIN MENU *******");

      printf("\n 1. Enter the first polynomial");

      printf("\n 2. Display the first polynomial");
      printf("\n 3. Enter the second polynomial");
      printf("\n 4. Display the second polynomial");
      printf("\n 5. Add the polynomials");
      printf("\n 6. Display the addition result");
      printf("\n 7. Subtract the polynomials");
      printf("\n 8. Display the subtraction result");
      printf("\n 9. EXIT");
      printf("\n\n Enter your option : ");
      scanf("%d", &option);

      switch(option) {
         case 1:
            do {
               printf("\n Enter the number: ");
               scanf("%d", &n);
               if(n == -1)
                  break;

               printf("\t Enter its coefficient: ");
               scanf("%d", &c);

               start1 = add_node(start1, n, c);
            } while(n != -1);
            break;

         case 2:
          display_poly(start1);
          break;

      case 3:
          do {
              printf("\n Enter the number: ");
              scanf("%d", &n);
              if(n == -1)
                  break;

              printf("\t Enter its coefficient: ");
              scanf("%d", &c);

              start2 = add_node(start2, n, c);
          } while(n != -1);
          break;

      case 4:
          display_poly(start2);
          break;

      case 5:
          start3 = add_poly(start1, start2, start3);
          printf("\nAddition Completed.");
          break;
```

```
            case 6:
                display_poly(start3);
                break;

            case 7:
                start4 = sub_poly(start1, start2, start4);
                printf("\nSubtraction Completed.");
                break;

            case 8:
                display_poly(start4);
                break;
        }
    } while(option != 9);

    return 0;
}

struct node* display_poly(struct node* start) {
    struct node* ptr = start;
    if(ptr == NULL) {
        printf("Polynomial is empty.\n");
        return start;
    }

    while(ptr != NULL) {
        printf("%d x^%d", ptr->num, ptr->coeff);
        if (ptr->next != NULL)
            printf(" + ");
        ptr = ptr->next;
    }
    printf("\n");
    return start;
}

struct node* add_poly(struct node* start1, struct node* start2, struct node* start3) {
    struct node *ptr1 = start1, *ptr2 = start2;
    int sum_num;

    while(ptr1 != NULL && ptr2 != NULL) {
        if(ptr1->coeff == ptr2->coeff) {
            sum_num = ptr1->num + ptr2->num;
            start3 = add_node(start3, sum_num, ptr1->coeff);
            ptr1 = ptr1->next;
            ptr2 = ptr2->next;
        } else if(ptr1->coeff > ptr2->coeff) {
            start3 = add_node(start3, ptr1->num, ptr1->coeff);
            ptr1 = ptr1->next;
        } else {
            start3 = add_node(start3, ptr2->num, ptr2->coeff);
            ptr2 = ptr2->next;
        }
    }

    while(ptr1 != NULL) {
        start3 = add_node(start3, ptr1->num, ptr1->coeff);
        ptr1 = ptr1->next;
    }

    while(ptr2 != NULL) {
        start3 = add_node(start3, ptr2->num, ptr2->coeff);
        ptr2 = ptr2->next;
    }

    return start3;
}

struct node* sub_poly(struct node* start1, struct node* start2, struct node*
start4) {
    struct node *ptr1 = start1, *ptr2 = start2;
    int sub_num;

    while(ptr1 != NULL && ptr2 != NULL) {
        if(ptr1->coeff == ptr2->coeff) {
            sub_num = ptr1->num - ptr2->num;
            start4 = add_node(start4, sub_num, ptr1->coeff);
            ptr1 = ptr1->next;
            ptr2 = ptr2->next;
        } else if(ptr1->coeff > ptr2->coeff) {
            start4 = add_node(start4, ptr1->num, ptr1->coeff);
            ptr1 = ptr1->next;
        } else {
            start4 = add_node(start4, -ptr2->num, ptr2->coeff);
            ptr2 = ptr2->next;
        }
    }

    while(ptr1 != NULL) {
        start4 = add_node(start4, ptr1->num, ptr1->coeff);
        ptr1 = ptr1->next;
    }

    while(ptr2 != NULL) {
        start4 = add_node(start4, -ptr2->num, ptr2->coeff);
        ptr2 = ptr2->next;
    }

    return start4;
}

struct node* add_node(struct node* start, int n, int c) {
    struct node* new_node = (struct node*)malloc(sizeof(struct node));
    new_node->num = n;
    new_node->coeff = c;
    new_node->next = NULL;
    if(start == NULL) {
        start = new_node;
    } else {
        struct node* ptr = start;
        while(ptr->next != NULL)
            ptr = ptr->next;
        ptr->next = new_node;
    }
    return start;
}
```

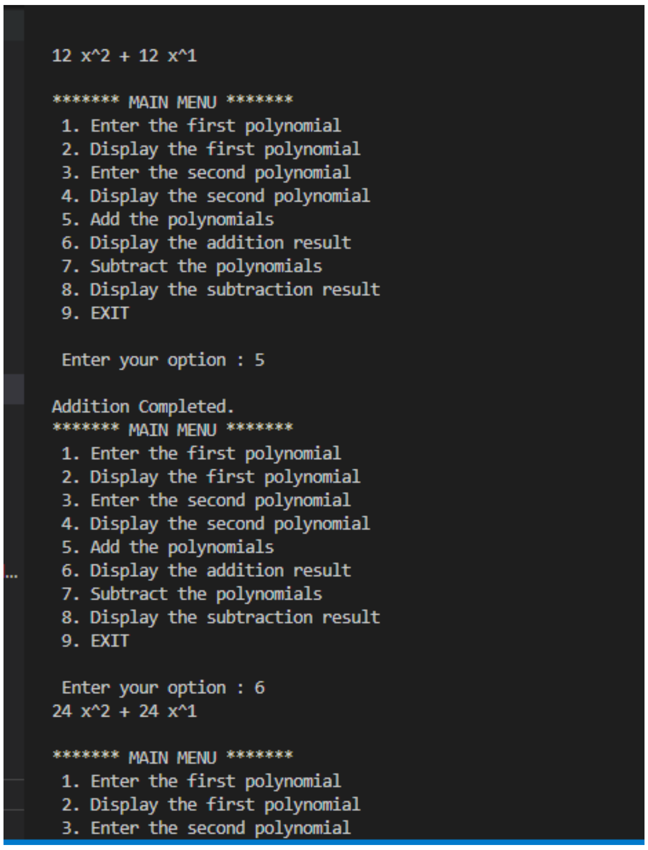
Output:



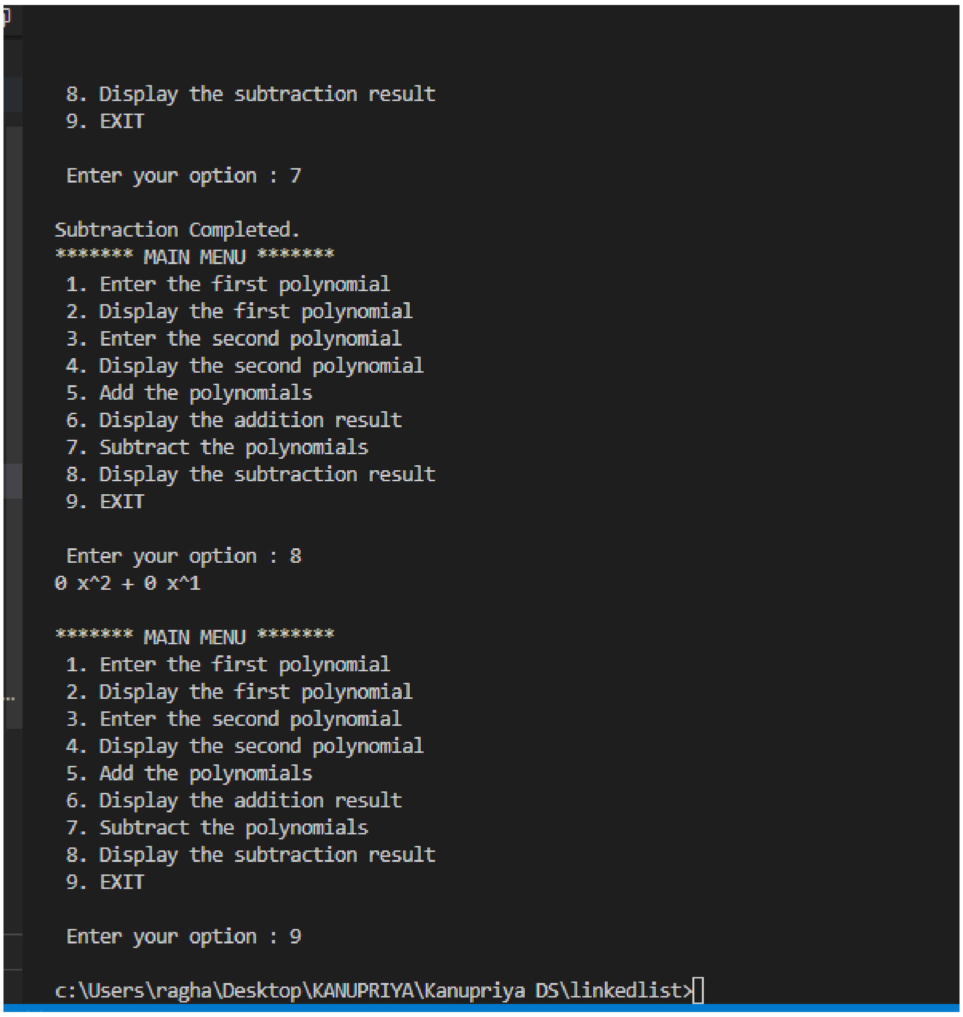
```
******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT

 Enter your option : 1

 Enter the number: 12
         Enter its coefficient: 2

 Enter the number: 12
         Enter its coefficient: 1

 Enter the number: -1

******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT

 Enter your option : 2
```



```
 Enter your option : 2
12 x^2 + 12 x^1

******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT

 Enter your option : 3

 Enter the number: 12
        Enter its coefficient: 2

 Enter the number: 12
        Enter its coefficient: 1

 Enter the number: -1

******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT
```



```
12 x^2 + 12 x^1

******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT

 Enter your option : 5

Addition Completed.
******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT

 Enter your option : 6
24 x^2 + 24 x^1

******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
```



```
 8. Display the subtraction result
 9. EXIT

 Enter your option : 7

Subtraction Completed.
******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT

 Enter your option : 8
0 x^2 + 0 x^1

******* MAIN MENU *******
 1. Enter the first polynomial
 2. Display the first polynomial
 3. Enter the second polynomial
 4. Display the second polynomial
 5. Add the polynomials
 6. Display the addition result
 7. Subtract the polynomials
 8. Display the subtraction result
 9. EXIT

 Enter your option : 9

c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\linkedlist>
```

EXPERIMENT 6:

AIM: Implementation of double ended queue using menu driven approach

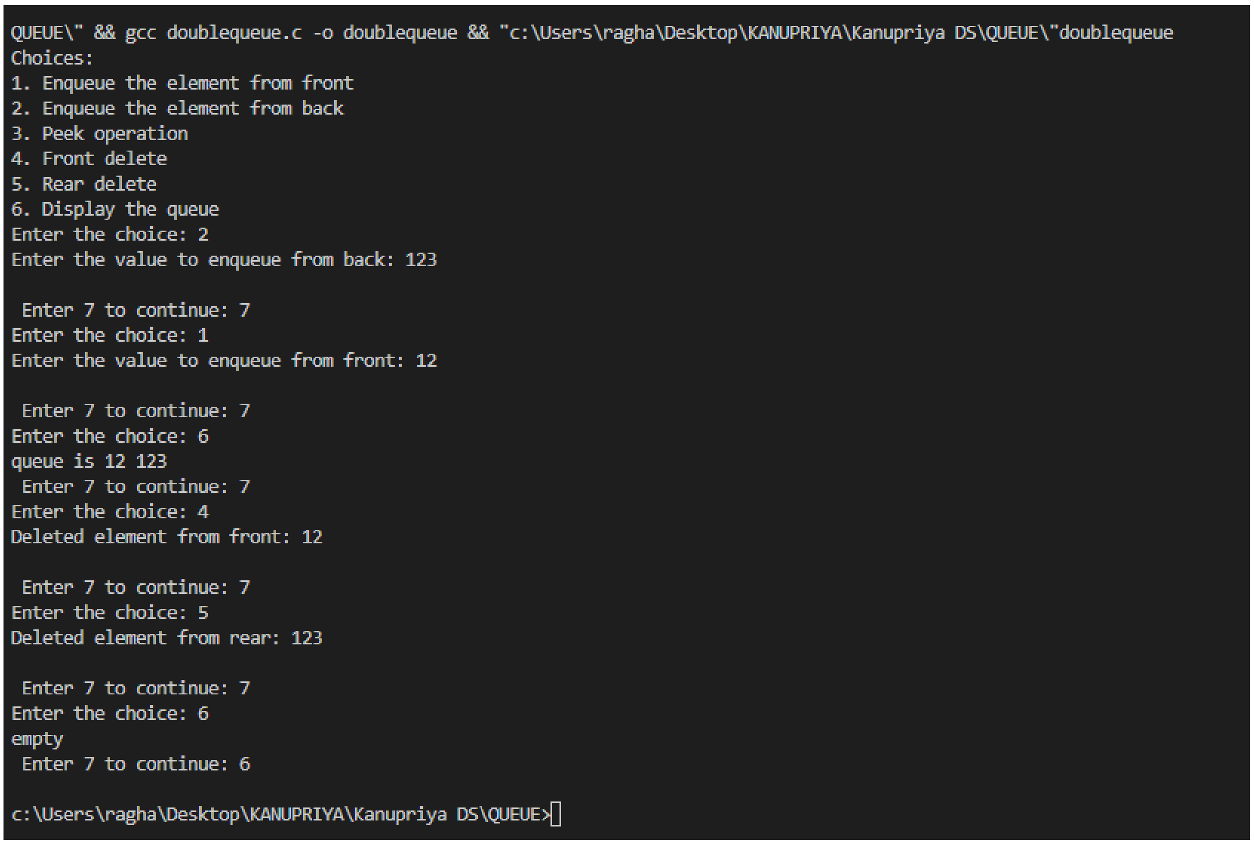
CODE:

```
#include <stdio.h>

#define MAX 20
int queue[MAX];
int rear = -1, front = -1;

void frontenqueue(int val) {
   if ((rear + 1) % MAX == front) {
      printf("Queue is full\n");
   } else if (front == -1) {
      front = rear = 0;
      queue[front] = val;
   } else if (front == 0) {
      front = MAX - 1;
      queue[front] = val;
   } else {
      front--;
      queue[front] = val;
   }
}

void rearenqueue(int val) {
   if ((rear + 1) % MAX == front) {
        printf("Queue is full\n");
    } else if (front == -1) {
        front = rear = 0;
        queue[rear] = val;
    } else if (rear == MAX - 1) {
        rear = 0;
        queue[rear] = val;
    } else {
        rear++;
        queue[rear] = val;
    }
}

void display(){int i=front;
    if(front==-1||rear==-1)
    printf("empty");    else{
        printf("queue is ");
        while(i!=rear){
            printf("%d ",queue[i]);
            i=(i+1)%MAX;


        }
        printf("%d",queue[rear]);
    }
}
```

```
int frontdelete() {
    if (front == -1) {
        printf("Queue is empty\n");
        return -1;
    } else {
        int temp = queue[front];
        if (front == rear) {
            front = rear = -1;
        } else {
            front = (front + 1) % MAX;
        }
        return temp;
    }
}

int reardelete() {
    if (front == -1) {
        printf("Queue is empty\n");
        return -1;
    } else {
        int temp = queue[rear];
        if (front == rear) {
            front = rear = -1;
        } else if (rear == 0) {
            rear = MAX - 1;
        } else {
            rear--;
        }
        return temp;
    }
}

void peek() {
    if (front == -1) {
        printf("Queue is empty\n");
    } else {
        printf("The first element is %d\n", queue[front]);
    }
}

int main() {
    int choice, e;
    printf("Choices: \n1. Enqueue the element from front\n2. Enqueue the
element from back\n3. Peek operation\n4. Front delete\n5. Rear delete\n6.
Display the queue\n");

    do {
        printf("Enter the choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the value to enqueue from front: ");
                int val;
                scanf("%d", &val);
                frontenqueue(val);
                break;
            case 2:
                printf("Enter the value to enqueue from back: ");
                int v;
                scanf("%d", &v);
                rearenqueue(v);
                break;
            case 3:
                peek();
                break;
            case 4:
                printf("Deleted element from front: %d\n", frontdelete());
                break;
            case 5:
                printf("Deleted element from rear: %d\n", reardelete());
                break;
            case 6:
                display();
                break;
            default:
                printf("Invalid choice. Please try again.\n");
                break;
        }
```

```
        printf("\n Enter 7 to continue: ");
        scanf("%d", &e);

    } while (e == 7);

    return 0;
}
```

Output:



```
QUEUE\" && gcc doublequeue.c -o doublequeue && "c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\QUEUE\"doublequeue
Choices:
1. Enqueue the element from front
2. Enqueue the element from back
3. Peek operation
4. Front delete
5. Rear delete
6. Display the queue
Enter the choice: 2
Enter the value to enqueue from back: 123

 Enter 7 to continue: 7
Enter the choice: 1
Enter the value to enqueue from front: 12

 Enter 7 to continue: 7
Enter the choice: 6
queue is 12 123
 Enter 7 to continue: 7
Enter the choice: 4
Deleted element from front: 12

 Enter 7 to continue: 7
Enter the choice: 5
Deleted element from rear: 123

 Enter 7 to continue: 7
Enter the choice: 6
empty
 Enter 7 to continue: 6

c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\QUEUE>
```

EXPERIMENT 7:

AIM: Implementation of BST using following operations – create, delete, display.

CODE:

```
#include <stdio.h>

#include <stdlib.h>


struct node {

   int data;

   struct node *left;

   struct node *right;

};


struct node *tree = NULL;


void create_tree(struct node *);

struct node *insertElement(struct node *, int);

void preorderTraversal(struct node *);

void inorderTraversal(struct node *);

void postorderTraversal(struct node *);


struct node *deleteElement(struct node *, int);


int main() {

   int option, val;

   struct node *ptr;
```

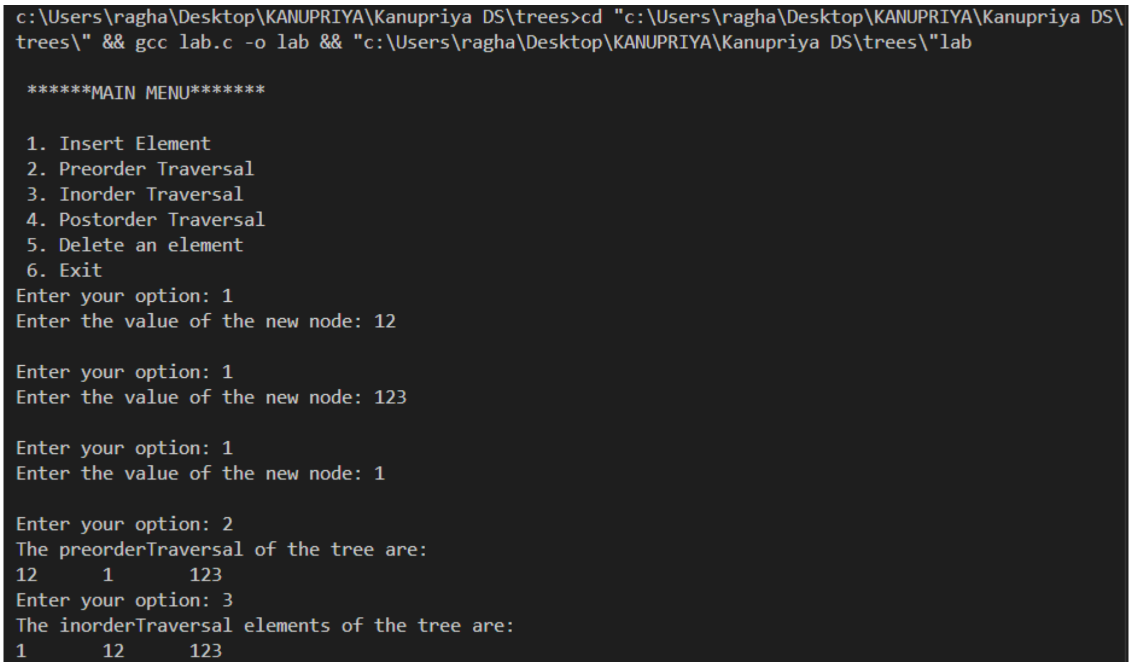
```
  create_tree(tree);
printf("\n ******MAIN MENU******* \n");
     printf("\n 1. Insert Element");
     printf("\n 2. Preorder Traversal");
     printf("\n 3. Inorder Traversal");
     printf("\n 4. Postorder Traversal");
     printf("\n 5. Delete an element");

     printf("\n 6. Exit");
  do {

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

     switch (option) {
        case 1:
           printf("Enter the value of the new node: ");
           scanf("%d", &val);
           tree = insertElement(tree, val);
           break;
        case 2:
           printf("The preorderTraversal of the tree are: \n");
           preorderTraversal(tree);
           break;
        case 3:
           printf("The inorderTraversal elements of the tree are: \n");
```

```
            inorderTraversal(tree);
            break;
        case 4:
            printf("The postorderTraversal elements of the tree are: \n");
            postorderTraversal(tree);
            break;

        case 5:
            printf("Enter the element to be deleted: ");
            scanf("%d", &val);
            tree = deleteElement(tree, val);
            break;

      }
   } while (option != 6);

   return 0;
}

void create_tree(struct node *tree) {
   tree = NULL;
}

struct node *insertElement(struct node *tree, int val) {
   struct node *ptr, *nodeptr, *parentptr;
   ptr = (struct node*)malloc(sizeof(struct node));
```

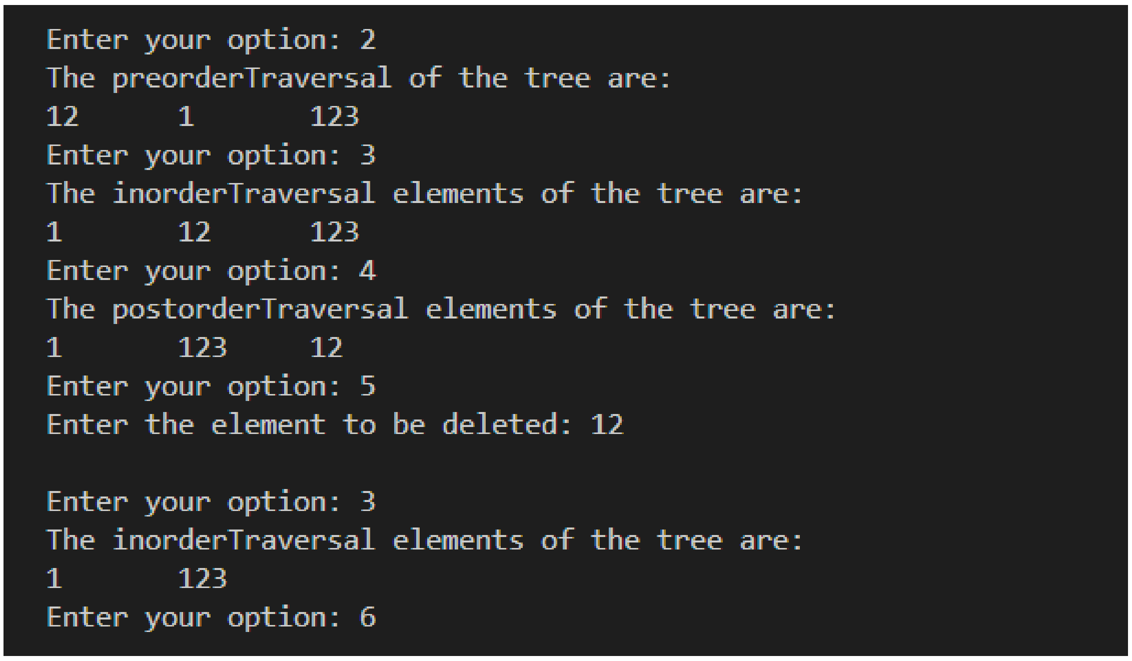
```
    ptr->data = val;

    ptr->left = NULL;

    ptr->right = NULL;


    if (tree == NULL) {

        tree = ptr;

    } else {

        parentptr = NULL;

        nodeptr = tree;

        while (nodeptr != NULL) {

            parentptr = nodeptr;

            if (val < nodeptr->data)

                nodeptr = nodeptr->left;

            else if(val >nodeptr->data)

                nodeptr = nodeptr->right;

                else {printf("Already data exists %d",val);

                return tree;}


        }

        if (val < parentptr->data)

            parentptr->left = ptr;

        else

            parentptr->right = ptr;

    }

    return tree;

}
```

```
void preorderTraversal(struct node *tree) {
   if (tree != NULL) {
      printf("%d\t", tree->data);
      preorderTraversal(tree->left);
      preorderTraversal(tree->right);  }}
void inorderTraversal(struct node *tree) {
   if (tree != NULL) {
      inorderTraversal(tree->left);
      printf("%d\t", tree->data);
      inorderTraversal(tree->right);}}
void postorderTraversal(struct node *tree) {
   if (tree != NULL) {
      postorderTraversal(tree->left);
      postorderTraversal(tree->right);
      printf("%d\t", tree->data);
   }
}
struct node *inOrderPredecessor(struct node* tree){
   tree = tree->left;
   while (tree->right!=NULL)
      tree = tree->right;
   return tree;
}

struct node *deleteElement(struct node *tree, int value){
   struct node* iPre;
   if (tree == NULL){
      return NULL;
   }
   if (tree->left==NULL&&tree->right==NULL){
      free(tree);
      return NULL;
   }
   if (value < tree->data){
      tree-> left = deleteElement(tree->left,value);
   }
   else if (value > tree->data){
      tree-> right = deleteElement(tree->right,value);
   }else{
 if (tree->left == NULL) {
         struct node* temp = tree->right;
         free(tree);
         return temp;
      } else if (tree->right == NULL) {
         struct node* temp = tree->left;
         free(tree);
         return temp; }
      iPre = inOrderPredecessor(tree);
      tree->data = iPre->data;
      tree->left = deleteElement(tree->left, iPre->data);}
   return tree;}
```

Output:



```
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\trees>cd "c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\
trees\" && gcc lab.c -o lab && "c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\trees\"lab

 ******MAIN MENU*******

 1. Insert Element
 2. Preorder Traversal
 3. Inorder Traversal
 4. Postorder Traversal
 5. Delete an element
 6. Exit
Enter your option: 1
Enter the value of the new node: 12

Enter your option: 1
Enter the value of the new node: 123

Enter your option: 1
Enter the value of the new node: 1

Enter your option: 2
The preorderTraversal of the tree are:
12      1       123
Enter your option: 3
The inorderTraversal elements of the tree are:
1       12      123
```



```
Enter your option: 2
The preorderTraversal of the tree are:
12      1       123
Enter your option: 3
The inorderTraversal elements of the tree are:
1       12      123
Enter your option: 4
The postorderTraversal elements of the tree are:
1       123     12
Enter your option: 5
Enter the element to be deleted: 12

Enter your option: 3
The inorderTraversal elements of the tree are:
1       123
Enter your option: 6
```

Experiment 8:

AIM: Implementation of Graph traversal using menu driven program (DFS & BFS).

CODE:

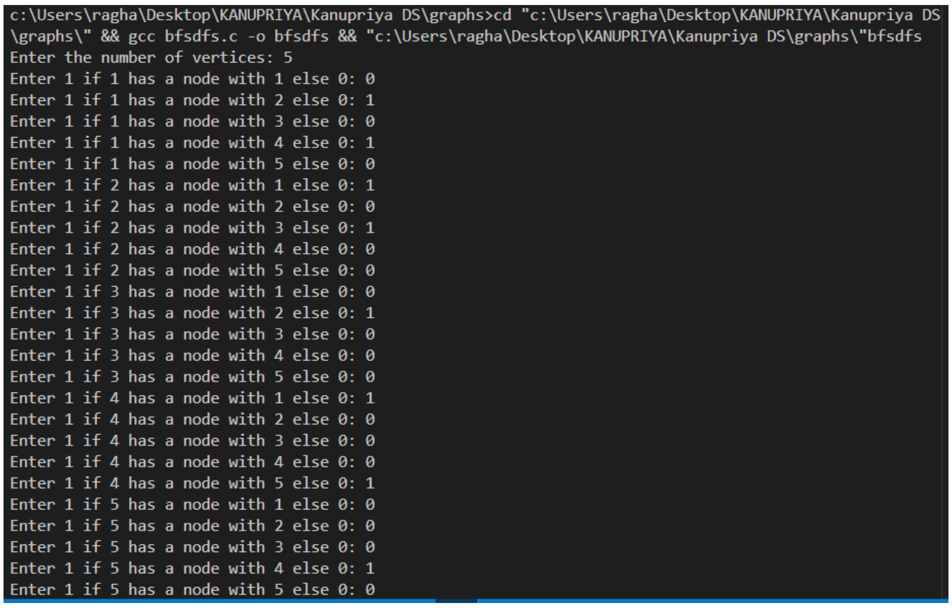
```
#include <stdio.h>

int q[20], front = -1, rear = -1, a[20][20], vis[20], stack[20];


int delete();

void add(int item);

void bfs(int s, int n);

void dfs(int s, int n);


int main()

{

   int n, i, s, ch, j;

   int c, dummy;

   printf("Enter the number of vertices: ");

   scanf("%d", &n);


   for (i = 1; i <= n; i++)

   {

      for (j = 1; j <= n; j++)

      {

         printf("Enter 1 if %d has a node with %d else 0: ", i, j);

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

      }

   }
```

```
    printf("The Adjacency Matrix is: \n");
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            printf(" %d", a[i][j]);
        }
        printf("\n");
    }

  do
{
    for (i = 1; i <= n; i++)
        vis[i] = 0;

    printf("\nMenu");
    printf("\n1. B.F.S");
    printf("\n2. D.F.S");
    printf("\nEnter the choice: ");
    scanf("%d", &ch);
    printf("Enter the source vertex: ");
    scanf("%d", &s);

    if (s < 1 || s > n)
    {
        printf("Invalid source vertex.\n");
        continue;
    }

    switch (ch)
    {
    case 1:
        bfs(s, n);
        break;
    case 2:
        dfs(s, n);
        break;
    default:
        printf("Invalid choice.\n");
    }

    printf("\nDo you want to continue? Press 3 for Yes: ");
    scanf("%d", &c);

} while (c == 3);

    return 0;
}

void bfs(int s, int n)
{
```

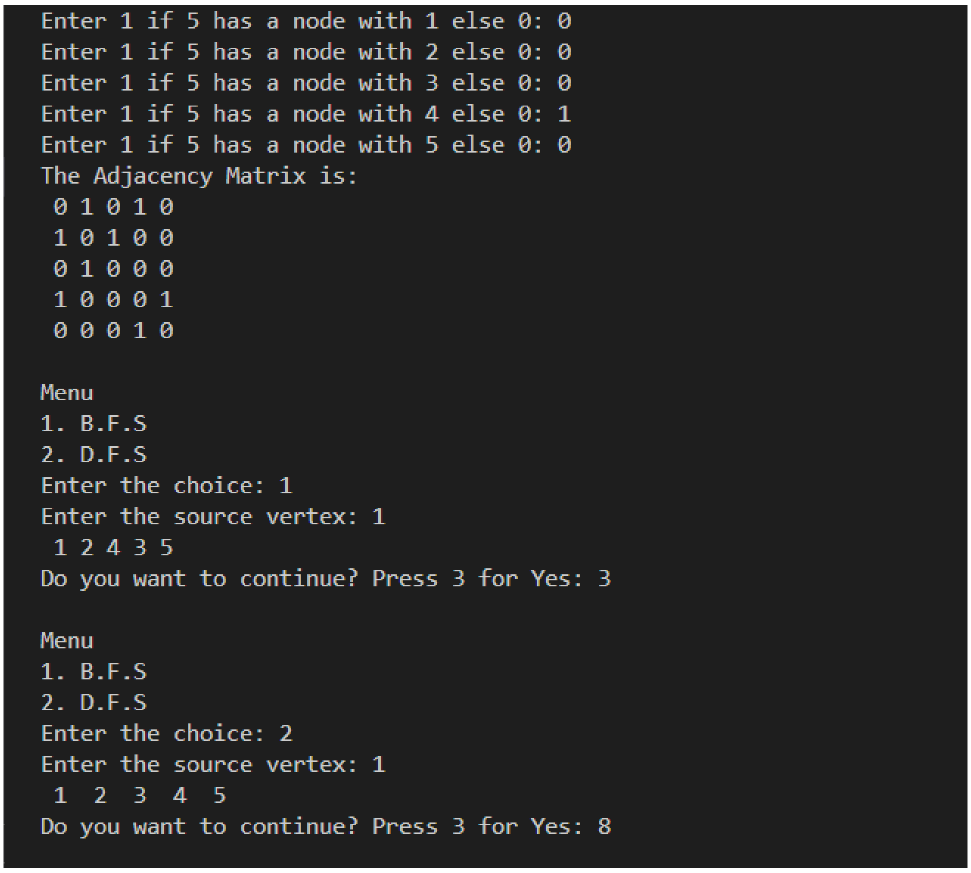
```
int p, i;
add(s);
vis[s] = 1;
p = delete();
if (p != 0)
    printf(" %d", p);

while (p != 0)
{
    for (i = 1; i <= n; i++)
    {
        if ((a[p][i] != 0) && (vis[i] == 0))
        {
            add(i);
            vis[i] = 1;
        }
    }

    p = delete();
    if (p != 0)
        printf(" %d", p);
}

for (i = 1; i <= n; i++)
    if (vis[i] == 0)
        bfs(i, n);
```

```
}

void add(int item)
{
    if (rear == 19)
        printf("Queue full..");
    else
    {
        if (rear == -1)
        {
            q[++rear] = item;
            front++;
        }
        else
            q[++rear] = item;
    }
}

int delete()
{
    int k;
    if ((front > rear) || (front == -1))
        return (0);
    else
    {
        k = q[front++];
        return (k);
    }
}

void dfs(int s, int n)
{int i;

        printf(" %d ", s);
   vis[s] = 1;

        for (i = 1; i <= n; i++)
        {
            if (a[s][i]==1&&(vis[i] == 0))
            {
              dfs(i,n);
            }
        }
}
```

OUTPUT:



```
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\graphs>cd "c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS
\graphs\" && gcc bfsdfs.c -o bfsdfs && "c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\graphs\"bfsdfs
Enter the number of vertices: 5
Enter 1 if 1 has a node with 1 else 0: 0
Enter 1 if 1 has a node with 2 else 0: 1
Enter 1 if 1 has a node with 3 else 0: 0
Enter 1 if 1 has a node with 4 else 0: 1
Enter 1 if 1 has a node with 5 else 0: 0
Enter 1 if 2 has a node with 1 else 0: 1
Enter 1 if 2 has a node with 2 else 0: 0
Enter 1 if 2 has a node with 3 else 0: 1
Enter 1 if 2 has a node with 4 else 0: 0
Enter 1 if 2 has a node with 5 else 0: 0
Enter 1 if 3 has a node with 1 else 0: 0
Enter 1 if 3 has a node with 2 else 0: 1
Enter 1 if 3 has a node with 3 else 0: 0
Enter 1 if 3 has a node with 4 else 0: 0
Enter 1 if 3 has a node with 5 else 0: 0
Enter 1 if 4 has a node with 1 else 0: 1
Enter 1 if 4 has a node with 2 else 0: 0
Enter 1 if 4 has a node with 3 else 0: 0
Enter 1 if 4 has a node with 4 else 0: 0
Enter 1 if 4 has a node with 5 else 0: 1
Enter 1 if 5 has a node with 1 else 0: 0
Enter 1 if 5 has a node with 2 else 0: 0
Enter 1 if 5 has a node with 3 else 0: 0
Enter 1 if 5 has a node with 4 else 0: 1
Enter 1 if 5 has a node with 5 else 0: 0
```



```
Enter 1 if 5 has a node with 1 else 0: 0
Enter 1 if 5 has a node with 2 else 0: 0
Enter 1 if 5 has a node with 3 else 0: 0
Enter 1 if 5 has a node with 4 else 0: 1
Enter 1 if 5 has a node with 5 else 0: 0
The Adjacency Matrix is:
 0 1 0 1 0
 1 0 1 0 0
 0 1 0 0 0
 1 0 0 0 1
 0 0 0 1 0

Menu
1. B.F.S
2. D.F.S
Enter the choice: 1
Enter the source vertex: 1
 1 2 4 3 5
Do you want to continue? Press 3 for Yes: 3

Menu
1. B.F.S
2. D.F.S
Enter the choice: 2
Enter the source vertex: 1
 1  2  3  4  5
Do you want to continue? Press 3 for Yes: 8
```

Experiement 9: WAP to implement Selection Sort, Insertion Sort, Quick Sort

Code: Insertion Sort

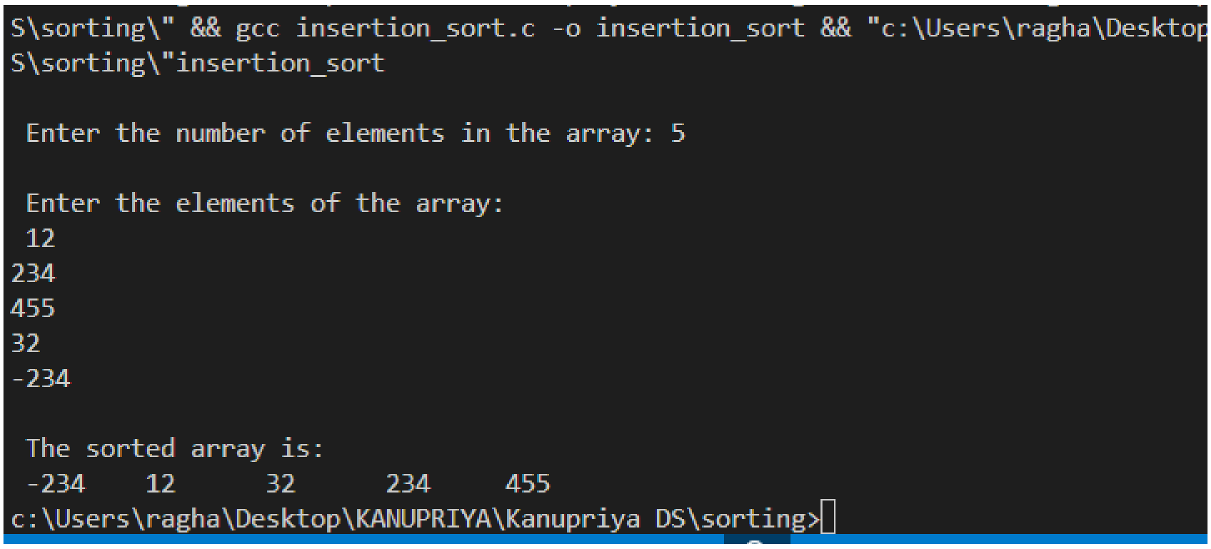
```
#include <stdio.h>

#define size 50
void insertion_sort(int arr[], int n);
void main()
{
int arr[size], i, n;
printf("\n Enter the number of elements in the array: ");
scanf("%d", &n);
printf("\n Enter the elements of the array:\n ");
for(i=0;i<n;i++)
scanf("%d", &arr[i]);
insertion_sort(arr, n);
printf("\n The sorted array is: \n");
for(i=0;i<n;i++)
printf(" %d\t", arr[i]);
}
void insertion_sort(int arr[], int n)
{int i, j, temp;
for(i=1;i<n;i++){
temp = arr[i];
j = i-1;
while((temp < arr[j]) && (j>=0)){
arr[j+1] = arr[j];
```

j--;}

arr[j+1] = temp;

}}

Output:



```
S\sorting\" && gcc insertion_sort.c -o insertion_sort && "c:\Users\ragha\Desktop
S\sorting\"insertion_sort

 Enter the number of elements in the array: 5

 Enter the elements of the array:
 12
234
455
32
-234

 The sorted array is:
 -234    12      32      234     455
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>
```

Quick Sort:

#include <stdio.h>

#define size 100

int partition(int a[], int beg, int end);

void quick_sort(int a[], int beg, int end);

void main()

{

int arr[size], i, n;

printf("\n Enter the number of elements in the array: ");

scanf("%d", &n);

printf("\n Enter the elements of the array: \n");

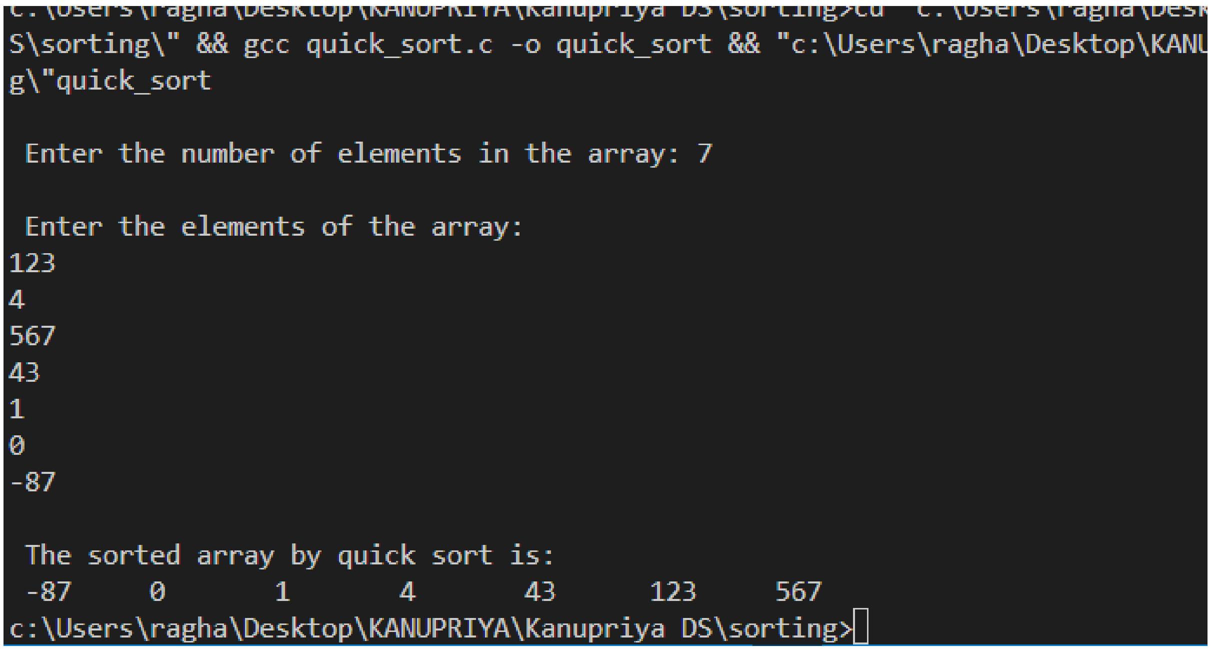
for(i=0;i<n;i++)

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

quick_sort(arr, 0, n-1);

```
printf("\n The sorted array by quick sort is: \n");
for(i=0;i<n;i++)
printf(" %d\t", arr[i]);}

int partition(int a[], int beg, int end){
int left, right, temp, loc, flag;
loc = left = beg;
right = end;
flag = 0;
while(flag != 1){
while((a[loc] <= a[right]) && (loc!=right))
right--;
if(loc==right)
flag =1;
else if(a[loc]>a[right])
{
temp = a[loc];
a[loc] = a[right];
a[right] = temp;
loc = right;
}
if(flag!=1)
{
while((a[loc] >= a[left]) && (loc!=left))
left++;
if(loc==left)
flag =1;
else if(a[loc] <a[left])
{
temp = a[loc];
a[loc] = a[left];
a[left] = temp;
loc = left;}}
}
return loc;
}
void quick_sort(int a[], int beg, int end){
int loc;
if(beg<end)
{
loc = partition(a, beg, end);
quick_sort(a, beg, loc-1);
quick_sort(a, loc+1, end);
}}
```

Output:



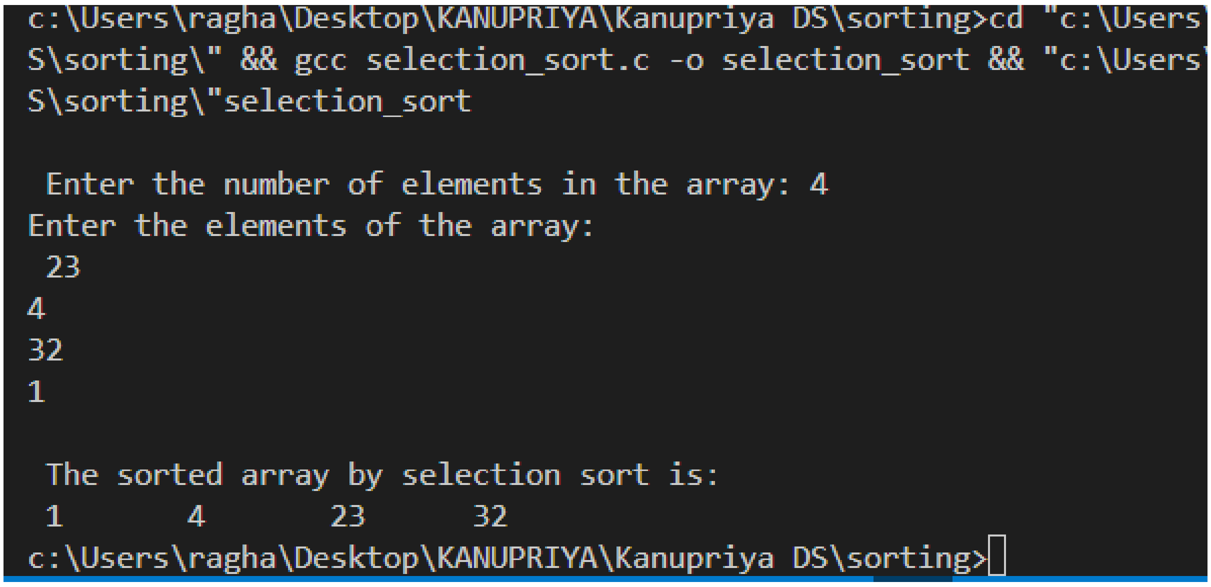
```
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>cd "c:\Users\ragha\Desk
S\sorting\" && gcc quick_sort.c -o quick_sort && "c:\Users\ragha\Desktop\KANU
g\"quick_sort

 Enter the number of elements in the array: 7

 Enter the elements of the array:
123
4
567
43
1
0
-87

 The sorted array by quick sort is:
 -87     0       1       4       43      123     567
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>
```

Selection Sort:

```
#include <stdio.h>

#include <stdlib.h>

int smallest(int arr[], int k, int n);

void selection_sort(int arr[], int n);

void main(int argc, char *argv[]) {

int arr[10], i, n;

printf("\n Enter the number of elements in the array: ");

scanf("%d", &n);

printf("Enter the elements of the array:\n ");

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

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

selection_sort(arr, n);

printf("\n The sorted array by selection sort is: \n");

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

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

}
```

```
int smallest(int arr[], int k, int n){
int pos = k, small=arr[k], i;
for(i=k+1;i<n;i++){
if(arr[i]< small){
small = arr[i];
pos = i;}}
return pos;}
void selection_sort(int arr[],int n){
int k, pos, temp;
for(k=0;k<n;k++){
pos = smallest(arr, k, n);
temp = arr[k];
arr[k] = arr[pos];
arr[pos] = temp;}
}
```

Output: Selection Sort:



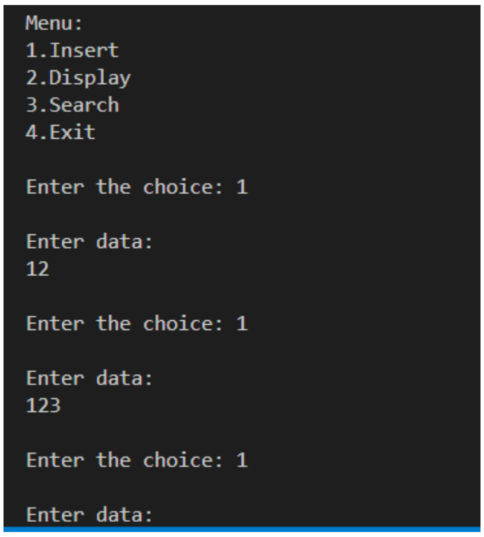
```
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>cd "c:\Users
S\sorting\" && gcc selection_sort.c -o selection_sort && "c:\Users
S\sorting\"selection_sort

 Enter the number of elements in the array: 4
Enter the elements of the array:
 23
4
32
1

 The sorted array by selection sort is:
 1       4       23      32
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>
```

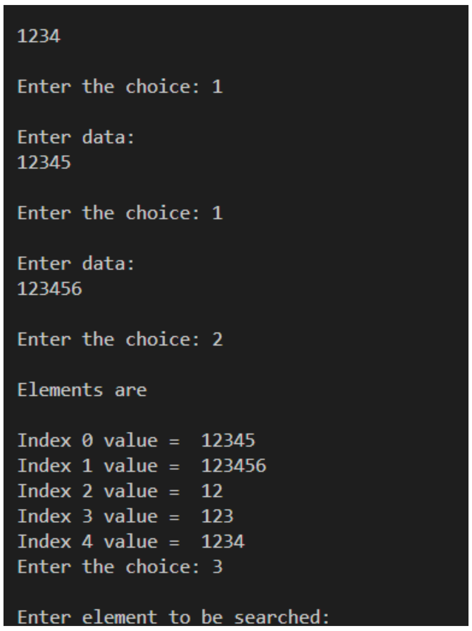
Experiment 10: Write a menu driven program in C to implement hashing techniques.

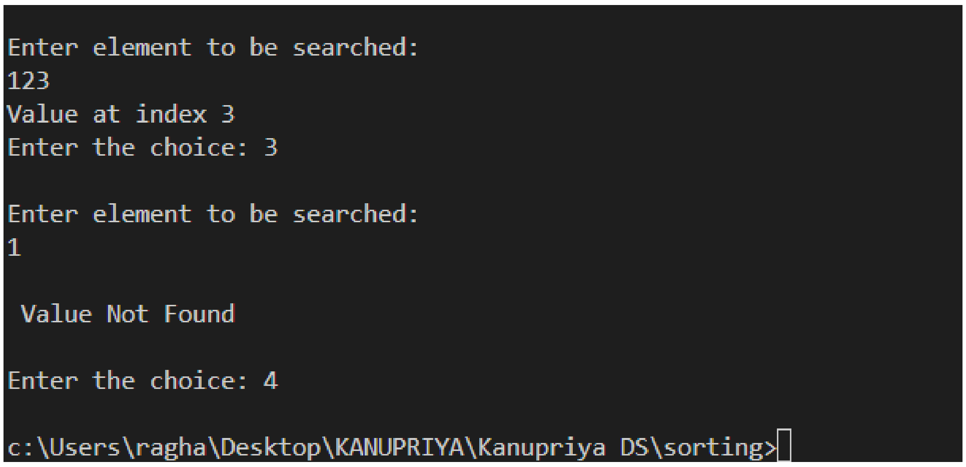
Code:

```
#include <stdio.h>

#include<stdlib.h>

#define TABLE_SIZE 5

int h[TABLE_SIZE] = {NULL};

void insert() {

 int key, index, i, hashingKey;

 printf("\nEnter data:\n");

 scanf("%d", &key);

 hashingKey = key % TABLE_SIZE;

 for(i = 0; i < TABLE_SIZE; i++){  index = (hashingKey + i) % TABLE_SIZE;

     if(h[index] == NULL){

        h[index] = key;

         break;}  }

    if(i == TABLE_SIZE){

     printf("\nelement cannot be inserted\n");}}

void search() {

 int key, index, i, hashingKey;

 printf("\nEnter element to be searched:\n");

 scanf("%d", &key);

 hashingKey = key % TABLE_SIZE;

 for(i = 0; i< TABLE_SIZE; i++) {

    index=(hashingKey + i) % TABLE_SIZE;

    if(h[index] == key) {

      printf("Value at index %d", index);
      break;}}
  if(i == TABLE_SIZE)
    printf("\n Value Not Found\n");}
void display() { int i;
  printf("\nElements are \n");
  for(i = 0; i < TABLE_SIZE; i++)
    printf("\nIndex %d value =  %d", i, h[i]);}
int main(){   int opt;
     printf("\nMenu:\n1.Insert\n2.Display\n3.Search\n4.Exit \n");
    while(1)
    {   printf("\nEnter the choice: ");
        scanf("%d", &opt);
        switch(opt)   {
            case 1:
                insert();
                break;
            case 2:
                display();
                break;
            case 3:
                search();
                break;
            case 4:exit(0);
            default:
            printf("Invalid");    } }
    return 0;}
```

Output:


```
Menu:
1.Insert
2.Display
3.Search
4.Exit

Enter the choice: 1

Enter data:
12

Enter the choice: 1

Enter data:
123

Enter the choice: 1

Enter data:
```


```
1234

Enter the choice: 1

Enter data:
12345

Enter the choice: 1

Enter data:
123456

Enter the choice: 2

Elements are

Index 0 value =  12345
Index 1 value =  123456
Index 2 value =  12
Index 3 value =  123
Index 4 value =  1234
Enter the choice: 3

Enter element to be searched:
```

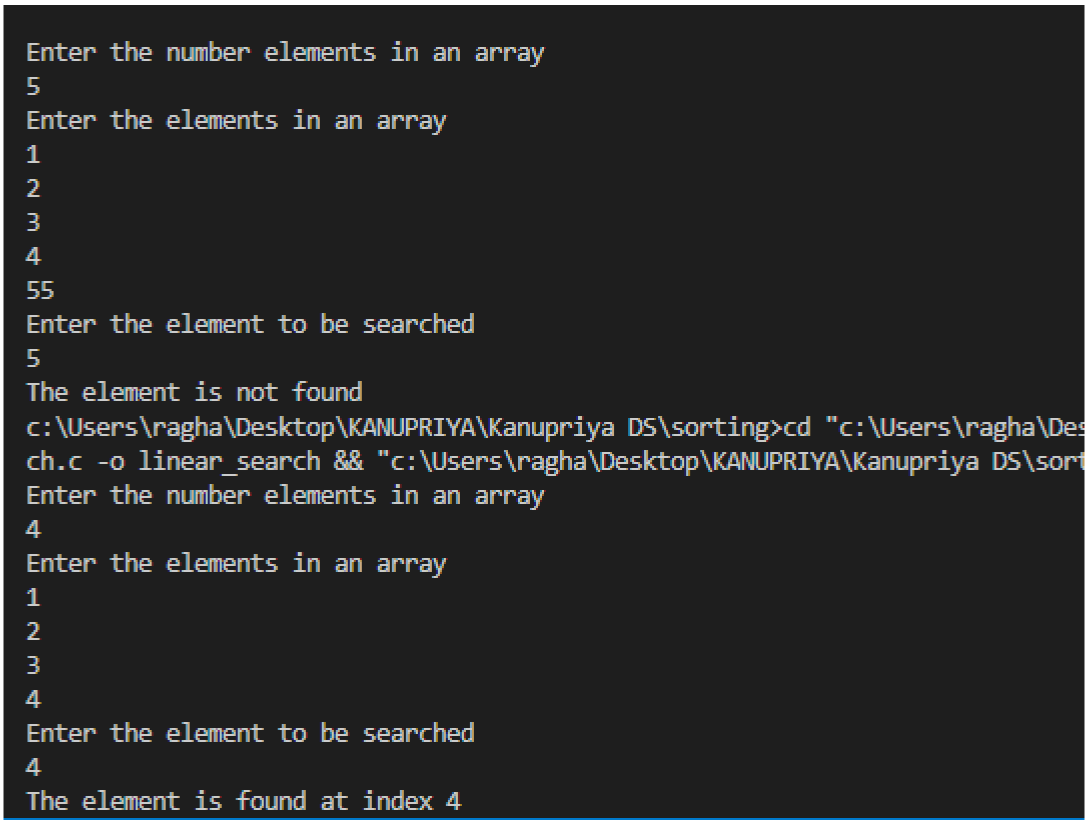

```
Enter element to be searched:
123
Value at index 3
Enter the choice: 3

Enter element to be searched:
1

 Value Not Found

Enter the choice: 4

c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>
```

Linear Search:

Code:

```
include<stdio.h>
#include<conio.h>
void main(){
   int n,i,temp,num;

printf("Enter the number elements in an array\n");
scanf("%d",&n);
printf("Enter the elements in an array\n");
int arr[n];
for(i=0;i<n;i++){
   scanf("%d",&arr[i]);
}
printf("Enter the element to be searched \n");
scanf("%d",&num);

for(i=0;i<n;i++){
      if(arr[i]==num){
 printf("The element is found at index %d ",i+1);break;}
  if(i==n-1){
 printf("The element is not found ");}

      }

}
```

Output:



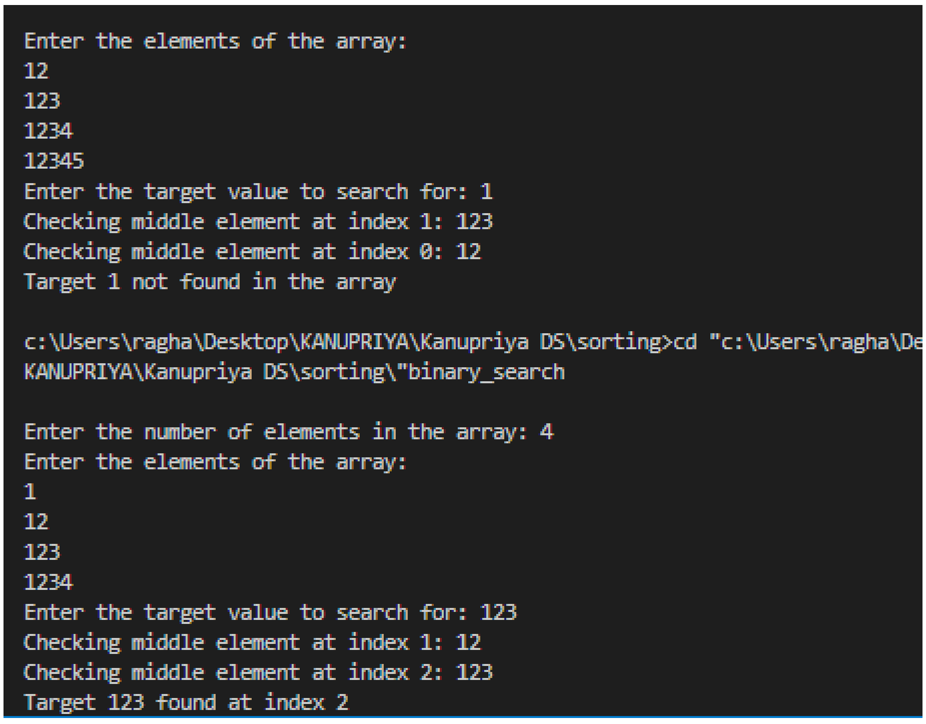
```
Enter the number elements in an array
5
Enter the elements in an array
1
2
3
4
55
Enter the element to be searched
5
The element is not found
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>cd "c:\Users\ragha\Des
ch.c -o linear_search && "c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sort
Enter the number elements in an array
4
Enter the elements in an array
1
2
3
4
Enter the element to be searched
4
The element is found at index 4
```

Binary Search:

Code:

```
#include <stdio.h>

void executeAtMid(int mid, int arr[], int size) {
    printf("Checking middle element at index %d: %d\n", mid, arr[mid]);
}

int binarySearch(int arr[], int size, int target) {
    int left = 0;
    int right = size - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;

        executeAtMid(mid, arr, size);

        if (arr[mid] == target) {
            return mid;
        }
        if (arr[mid] < target) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1;
}

int main() {
    int n, i;
    printf("\nEnter the number of elements in the array: ");
    scanf("%d", &n);

    int arr[n];  // This works in C99 or later, but the size must be determined at
runtime.

    printf("Enter the elements of the array:\n");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int target;
    printf("Enter the target value to search for: ");
    scanf("%d", &target);

    int result = binarySearch(arr, n, target);

    if (result != -1) {
        printf("Target %d found at index %d\n", target, result);
    } else {
        printf("Target %d not found in the array\n", target);
    }    return 0;
}
```

Output:



```
Enter the elements of the array:
12
123
1234
12345
Enter the target value to search for: 1
Checking middle element at index 1: 123
Checking middle element at index 0: 12
Target 1 not found in the array

c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>cd "c:\Users\ragha\De
KANUPRIYA\Kanupriya DS\sorting\"binary_search

Enter the number of elements in the array: 4
Enter the elements of the array:
1
12
123
1234
Enter the target value to search for: 123
Checking middle element at index 1: 12
Checking middle element at index 2: 123
Target 123 found at index 2
```

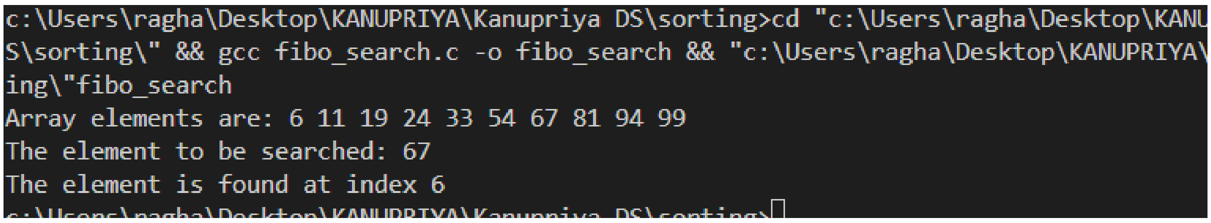
Fibonacci Search:

Code:

```
#include <stdio.h>
int min(int, int);
int fibonacci_search(int[], int, int);
int min(int a, int b){
   return (a > b) ? b : a;}
int fibonacci_search(int arr[], int n, int key){
   int offset = -1;
   int Fm2 = 0;
   int Fm1 = 1;
   int Fm = Fm2 + Fm1;
   while (Fm < n) {
      Fm2 = Fm1;
      Fm1 = Fm;
      Fm = Fm2 + Fm1; }
   while (Fm > 1) {
      int i = min(offset + Fm2, n - 1);
      if (arr[i] < key) {
         Fm = Fm1;
         Fm1 = Fm2;
         Fm2 = Fm - Fm1;
         offset = i;
      } else if (arr[i] > key) {
         Fm = Fm2;
         Fm1 = Fm1 - Fm2;
```

```
            Fm2 = Fm - Fm1;
        } else
            return i;   }
    if (Fm1 && arr[offset + 1] == key)
        return offset + 1;
    return -1;
}
int main(){
   int i, n, key, pos;
   int arr[10] = {6, 11, 19, 24, 33, 54, 67, 81, 94, 99};
   printf("Array elements are: ");
   int len = sizeof(arr) / sizeof(arr[0]);
   for(int j = 0; j<len; j++){
      printf("%d ", arr[j]);
   } n = 10;
   key = 67;
   printf("\nThe element to be searched: %d", key);
   pos = fibonacci_search(arr, n, key);
    if(pos >= 0)
        printf("\nThe element is found at index %d", pos);
    else
        printf("\nUnsuccessful Search");}
```

Output:



```
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>cd "c:\Users\ragha\Desktop\KANU
S\sorting\" && gcc fibo_search.c -o fibo_search && "c:\Users\ragha\Desktop\KANUPRIYA\
ing\"fibo_search
Array elements are: 6 11 19 24 33 54 67 81 94 99
The element to be searched: 67
The element is found at index 6
c:\Users\ragha\Desktop\KANUPRIYA\Kanupriya DS\sorting>
```