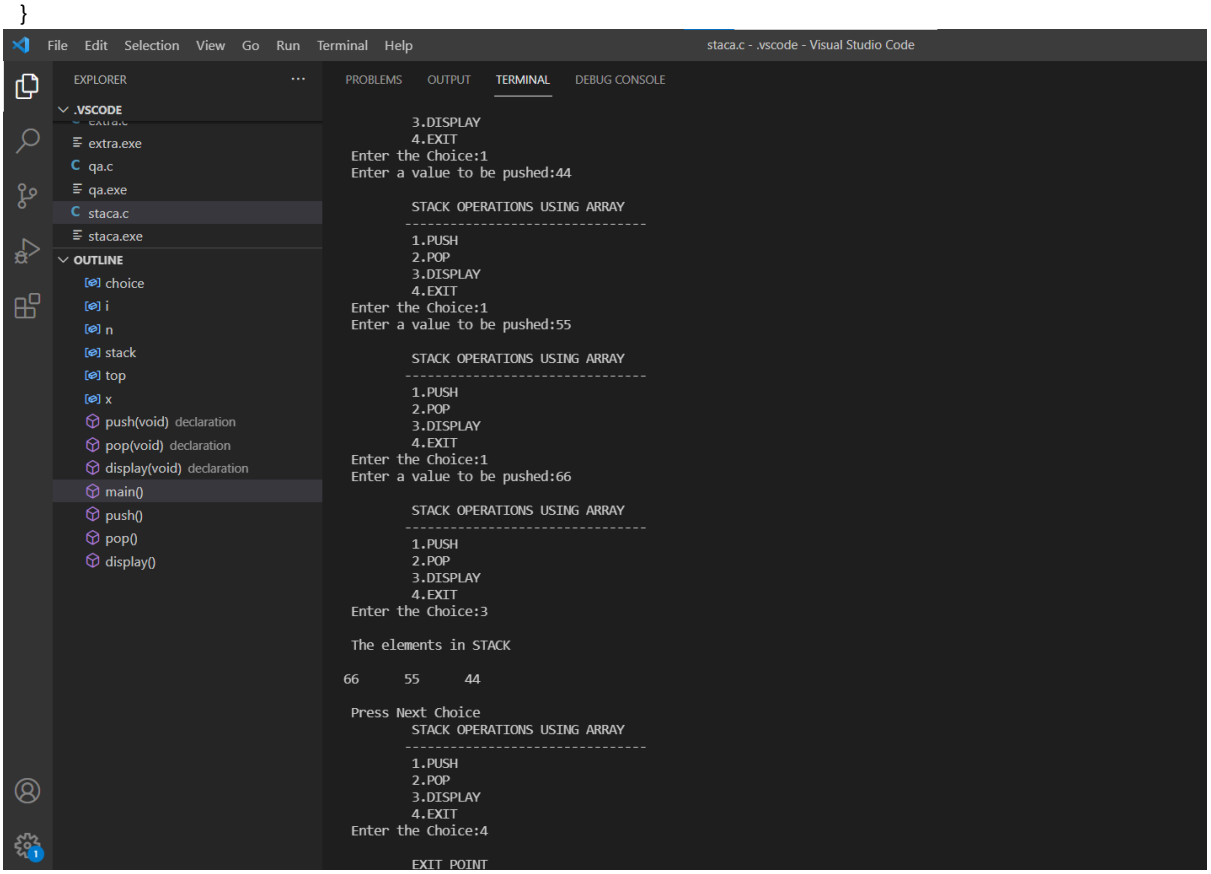
Implement stack using arrays

```
#include<stdio.h>
int stack[100],choice,n,top,x,i;
void push(void);
void pop(void);
void display(void);
int main()
{

   top=-1;

   printf("\n Enter the size of STACK[MAX=100]:");
   scanf("%d",&n);
   do
   {
   printf("\n\t STACK OPERATIONS USING ARRAY");
   printf("\n\t--------------------------------");
   printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");


      printf("\n Enter the Choice:");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         {
            push();
            break;
         }
         case 2:
         {
            pop();
            break;
         }
         case 3:
         {
            display();
            break;
         }
         case 4:
         {
            printf("\n\t EXIT POINT ");
            break;
         }
         default:
         {
            printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
         }

      }
   }
  while (choice!=4);
    return 0;
}
void push()
{
   if(top>=n-1)
 {
      printf("\n\tSTACK is over flow");

   }
   else
```

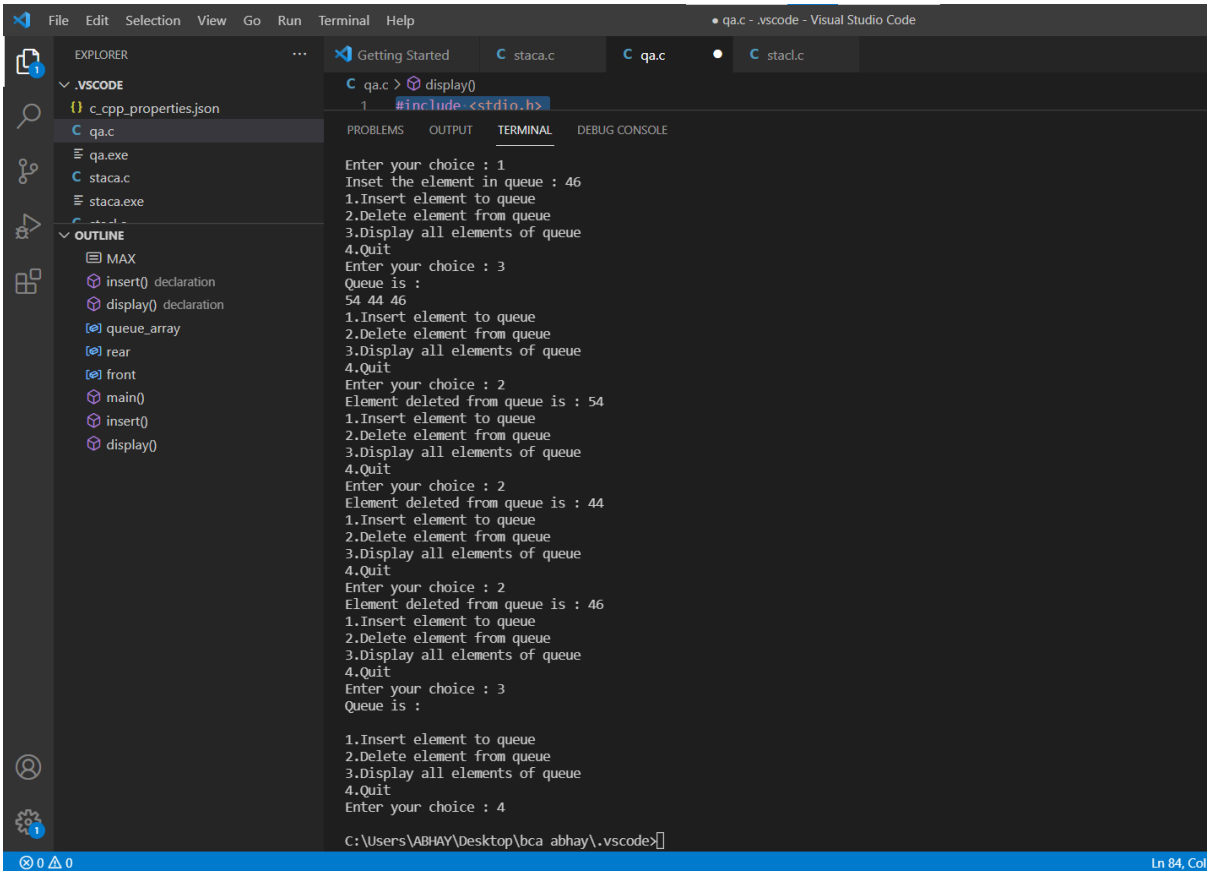
```
  {
    printf(" Enter a value to be pushed:");
    scanf("%d",&x);
    top++;
    stack[top]=x;
  }
}
void pop()
{
  if(top<=-1)
  {
    printf("\n\t Stack is under flow");
  }
  else
  {
    printf("\n\t The popped elements is %d",stack[top]);
    top--;
  }
}
void display()
{
  if(top>=0)
  {
    printf("\n The elements in STACK \n\n");
    for(i=top; i>=0; i--)
      printf("%d\t",stack[i]);
    printf("\n\n Press Next Choice");
  }
  else
  {
    printf("\n The STACK is empty");
  }
 }
```



Implement queues using arrays

```
#include <stdio.h>

#define MAX 50

void insert();
void delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
main()
{
   int choice;
   while (1)
   {
      printf("1.Insert element to queue \n");
      printf("2.Delete element from queue \n");
      printf("3.Display all elements of queue \n");
      printf("4.Quit \n");
      printf("Enter your choice : ");
      scanf("%d", &choice);
      switch (choice)
      {
         case 1:
         insert();
         break;
         case 2:
         delete();
         break;
         case 3:
         display();
         break;
         case 4:
         exit(1);
         default:
         printf("Wrong choice \n");
      }
   }
}

void insert()
{
   int add_item;
   if (rear == MAX - 1)
   printf("Queue Overflow \n");
   else
   {
      if (front == - 1)

      front = 0;
      printf("Inset the element in queue : ");
      scanf("%d", &add_item);
      rear = rear + 1;
      queue_array[rear] = add_item;
   }
}

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

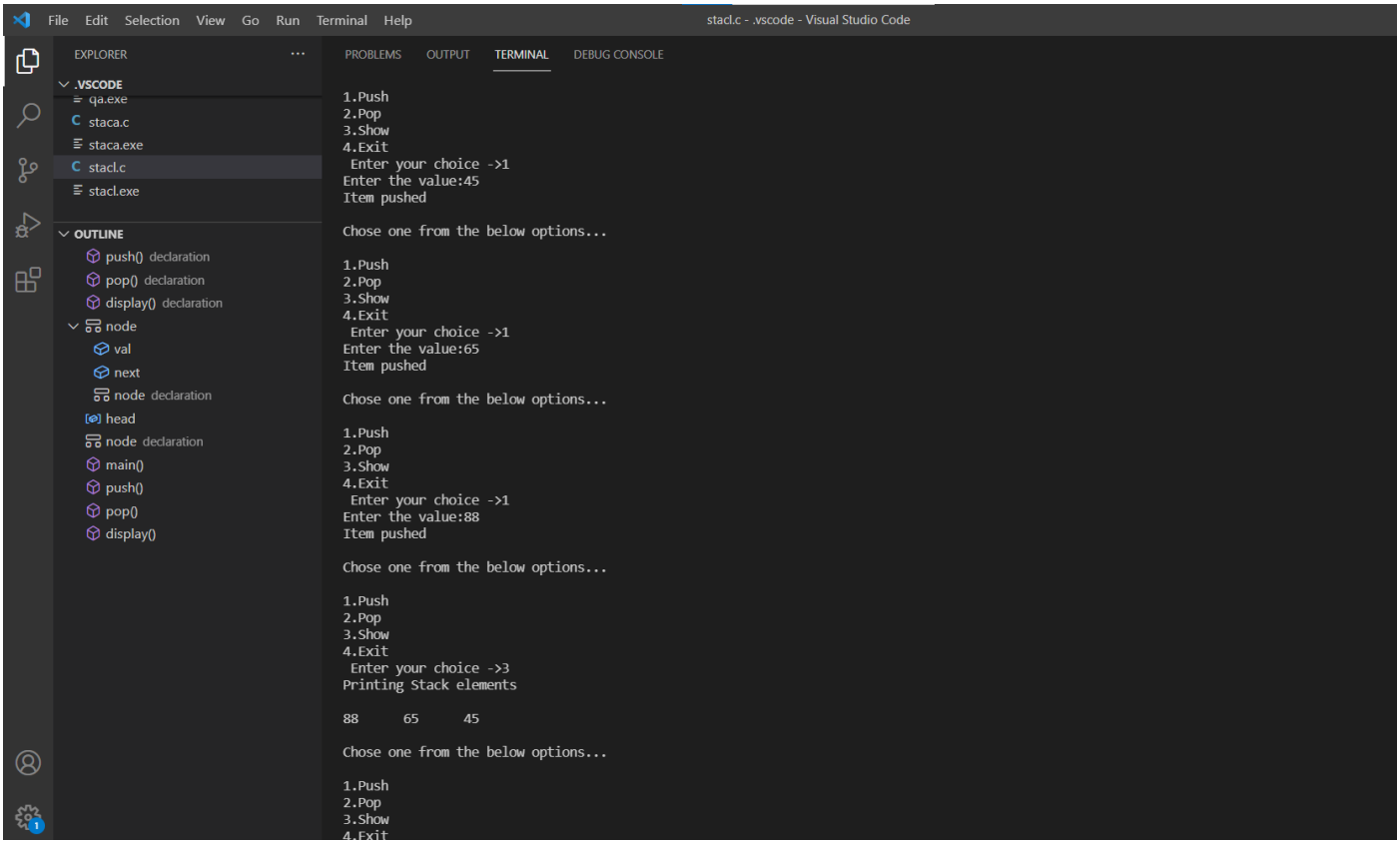
```
    return ;
  }
  else
  {
    printf("Element deleted from queue is : %d\n", queue_array[front]);
    front = front + 1;
  }
}
 void display()
{
  int i;
  if (front == - 1)
    printf("Queue is empty \n");
  else
  {
    printf("Queue is : \n");
    for (i = front; i <= rear; i++)
      printf("%d ", queue_array[i]);
    printf("\n");
  }
}
```



Implement stack using linked list

```
#include <stdio.h>
#include <stdlib.h>
void push();
void pop();
void display();
struct node
{
int val;
struct node *next;
};
struct node *head;

void main ()
{
   int choice=0;
   printf("\n*********Stack operations using linked list*********\n");
   printf("\n----------------------------------------------\n");
   while(choice != 4)
   {
      printf("\n\nChose one from the below options...\n");
      printf("\n1.Push\n2.Pop\n3.Show\n4.Exit");
      printf("\n Enter your choice ->");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
         {
            push();
            break;
         }
         case 2:
         {
            pop();
            break;
         }
         case 3:
         {
            display();
            break;
         }
         case 4:
         {
            printf("Exiting....");
            break;
         }
         default:
         {
            printf("Please Enter valid choice--> ");
         }
   };
}
}
void push ()
{
   int val;
   struct node *ptr = (struct node*)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
      printf("not able to push the element");
   }
   else
   {
```

```
      printf("Enter the value:");
      scanf("%d",&val);
      if(head==NULL)
      {
         ptr->val = val;
         ptr -> next = NULL;
         head=ptr;
      }
      else
      {
         ptr->val = val;
         ptr->next = head;
         head=ptr;

      }
      printf("Item pushed");

   }
}

void pop()
{
   int item;
   struct node *ptr;
   if (head == NULL)
   {
      printf("Underflow");
   }
   else
   {
      item = head->val;
      ptr = head;
      head = head->next;
      free(ptr);
      printf("Item popped");

   }
}
void display()
{
   int i;
   struct node *ptr;
   ptr=head;
   if(ptr == NULL)
   {
      printf("Stack is empty\n");
   }
   else
   {
      printf("Printing Stack elements \n\n");
      while(ptr!=NULL)
      {
         printf("%d\t",ptr->val);
         ptr = ptr->next;
      }
   }
}
```

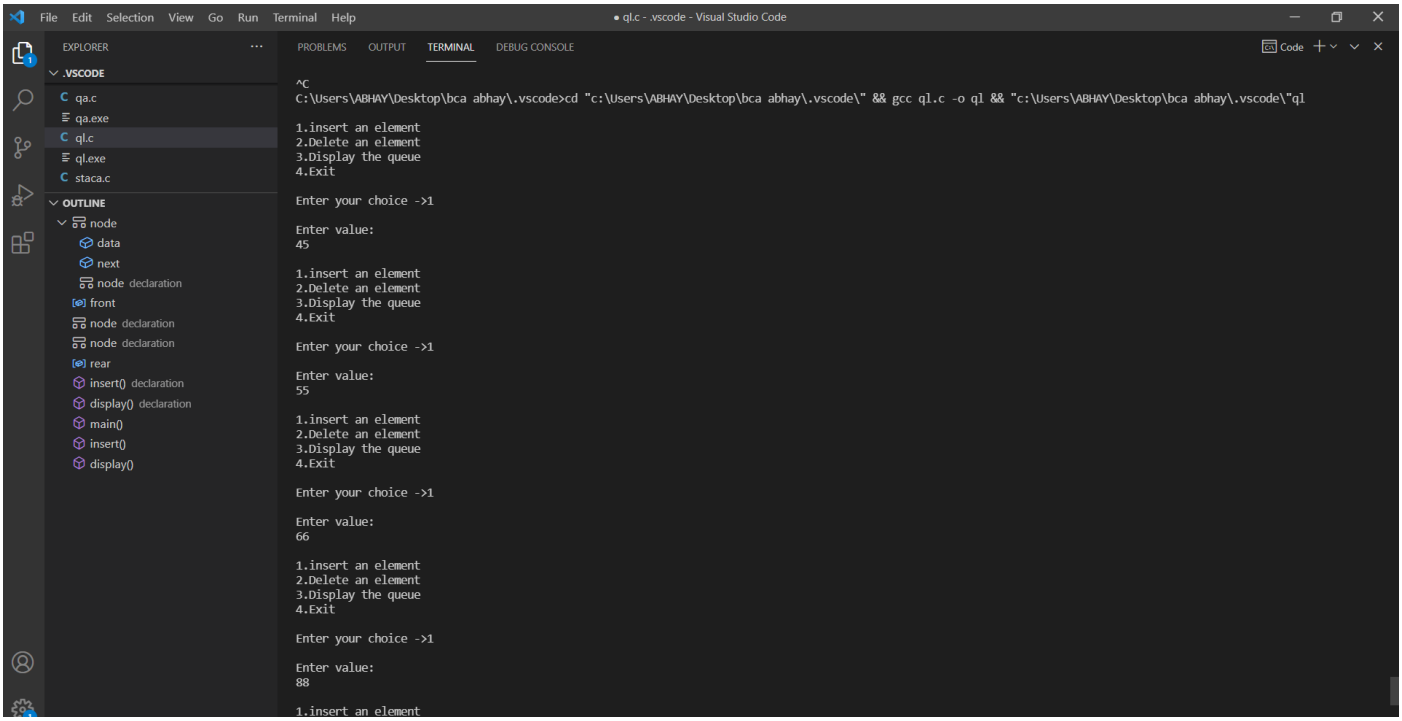

File Edit Selection View Go Run Terminal Help
stacl.c - .vscode - Visual Studio Code
EXPLORER
.VSCODE
qa.exe
staca.c
staca.exe
stacl.c
stacl.exe
OUTLINE
push() declaration
pop() declaration
display() declaration
node
val
next
node declaration
head
node declaration
main()
push()
pop()
display()
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
1.Push
2.Pop
3.Show
4.Exit
Enter your choice ->1
Enter the value:45
Item pushed
Chose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice ->1
Enter the value:65
Item pushed
Chose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice ->1
Enter the value:88
Item pushed
Chose one from the below options...
1.Push
2.Pop
3.Show
4.Exit
Enter your choice ->3
Printing Stack elements
88 65 45
Chose one from the below options...
1.Push
2.Pop
3.Show
4.Exit

Implement queues using linked list

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
   int data;
   struct node *next;
};
struct node *front;
struct node *rear;
void insert();
void delete();
void display();
void main ()
{
   int choice;
   while(choice != 4)
   {


      printf("\n\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");
      printf("\nEnter your choice ->");
      scanf("%d",& choice);
      switch(choice)
      {
         case 1:
         insert();
         break;
         case 2:
         delete();
         break;
         case 3:
         display();
         break;
         case 4:
         exit(0);
         break;
         default:
         printf("\nEnter valid choice??\n");
      }
   }
}
void insert()
{
   struct node *ptr;
   int item;

   ptr = (struct node *) malloc (sizeof(struct node));
   if(ptr == NULL)
   {
      printf("\nOVERFLOW\n");
      return;
   }
   else
   {
      printf("\nEnter value:\n");
      scanf("%d",&item);
      ptr -> data = item;
      if(front == NULL)
      {
         front = ptr;
         rear = ptr;
         front -> next = NULL;
```

```
            rear -> next = NULL;
        }
        else
        {
            rear -> next = ptr;
            rear = ptr;
            rear->next = NULL;
        }
    }
}
void delete ()
{
    struct node *ptr;
    if(front == NULL)
    {
        printf("\nUNDERFLOW\n");
        return;
    }
    else
    {
        ptr = front;
        front = front -> next;
        free(ptr);
    }
}
void display()
{
    struct node *ptr;
    ptr = front;
    if(front == NULL)
    {
        printf("\nEmpty queue\n");
    }
    else
    {   printf("\nprinting values .....\n");
        while(ptr != NULL)
        {
            printf("%d\t",ptr -> data);
            ptr = ptr -> next;
        }
    }
```

}



Implement linear search using linked lis

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


struct node
{
   int data;
   struct node *next;
} * head;



void createList(int n);
void displayList();
int  search(int key);


int main()
{
   int n, keyToSearch, index;


   printf("Enter number of node to create: ");
   scanf("%d", &n);

   createList(n);


   printf("\nData in list: \n");
   displayList();


   printf("\nEnter element to search: ");
   scanf("%d", &keyToSearch);


   index = search(keyToSearch);
```

```
    if (index >= 0)
        printf("%d found in the list at position %d\n", keyToSearch, index + 1);
    else
        printf("%d not found in the list.\n", keyToSearch);


    return 0;
}


void createList(int n)
{
    struct node *newNode, *temp;
    int data, i;

    head = malloc(sizeof(struct node));


    if (head == NULL)
    {
        printf("Unable to allocate memory. Exiting from app.");
        exit(0);
    }



    printf("Enter data of node 1: ");
    scanf("%d", &data);

    head->data = data;
    head->next = NULL;
    temp = head;


    for (i = 2; i <= n; i++)
    {
        newNode = malloc(sizeof(struct node));


        if (newNode == NULL)
        {
            printf("Unable to allocate memory. Exiting from app.");
            exit(0);
        }

        printf("Enter data of node %d: ", i);
        scanf("%d", &data);

        newNode->data = data;
        newNode->next = NULL;

        temp->next = newNode;
        temp = temp->next;
    }

}


void displayList()
{
    struct node *temp;
```

```
    if (head == NULL)
    {
        printf("List is empty.\n");
        return;
    }

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


int search(int key)
{
    int index;
    struct node *curNode;

    index = 0;
    curNode = head;


    while (curNode != NULL && curNode->data != key)
    {
        index++;
        curNode = curNode->next;
    }


    return (curNode != NULL) ? index : -1;
}
```