Neeraj Kumar [21MCF1R27]

DS Lab Assignment : 4 [20-02-2022]

Tasks : [1 to 10]

# Program 1. Write a program to implement stack using arrays.

# Source Code

*#include*<iostream>

*using* *namespace* std;

*int* *stack*[1000],*n*,*top=-*1;

*void* *push*();

*void* *pop*();

*void* *display*();

*int* *main*()

{

*int* *ch*;

*cout<<*"Enter the size of Stack:";

*cin>>n*;

*cout<<*"Attention!,Now You can only insert "*<<n<<*" elements in the stack"*<<endl*;

*cout<<*"\t\t\*\*\*\*\*\*\*\*\*\*\*Stack Menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"*<<endl*;

*cout<<*"\t1).Enter 1 to Push"*<<endl*;

*cout<<*"\t2).Enter 2 to Pop"*<<endl*;

*cout<<*"\t3).Enter 3 to Display"*<<endl*;

*cout<<*"\t4).Press 0 to exit Stack Menu"*<<endl*;

*do*{

*cout<<endl<<*"Please,Make Your Choice:";

*cin>>ch*;

*switch*(*ch*)

        {

*case* 1:

*push*();

*break*;

*case* 2:

*pop*();

*break*;

*case* 3:

*display*();

*break*;

*case* 0:

            {

*cout<<*"Exiting from the stack....."*<<endl*;

*exit*(0);

            }

*break*;

*default*:

*cout<<*"Invalid Choice";

        }

    }*while*(*ch!=*0);

*return* 0;

}

*void* *push*()

{

*if*(*top>=n-*1)

*cout<<*"Stack Overflow";

*else*{

*int* *x*;

*cout<<*"Enter an element to push:";

*cin>>x*;

*top++*;

*stack*[*top*]*=x*;

*cout<<stack*[*top*]*<<*" inserted into Stack";

    }

}

*void* *pop*()

{

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

*cout<<*"Stack Underflow";

*else*

    {

*cout<<stack*[*top*]*<<*" removed from the stack";

*top--*;

    }

}

*void* *display*()

{

*if*(*top>=*0)

    {

*cout<<*"-----------------------------------------"*<<endl*;

*cout<<*"Stack Elements:";

*for*(*int* *i=top*;*i>=*0;*i--*)

        {

*cout<<stack*[*i*]*<<*" ";

        }

*cout<<endl<<*"------------------------------------------";

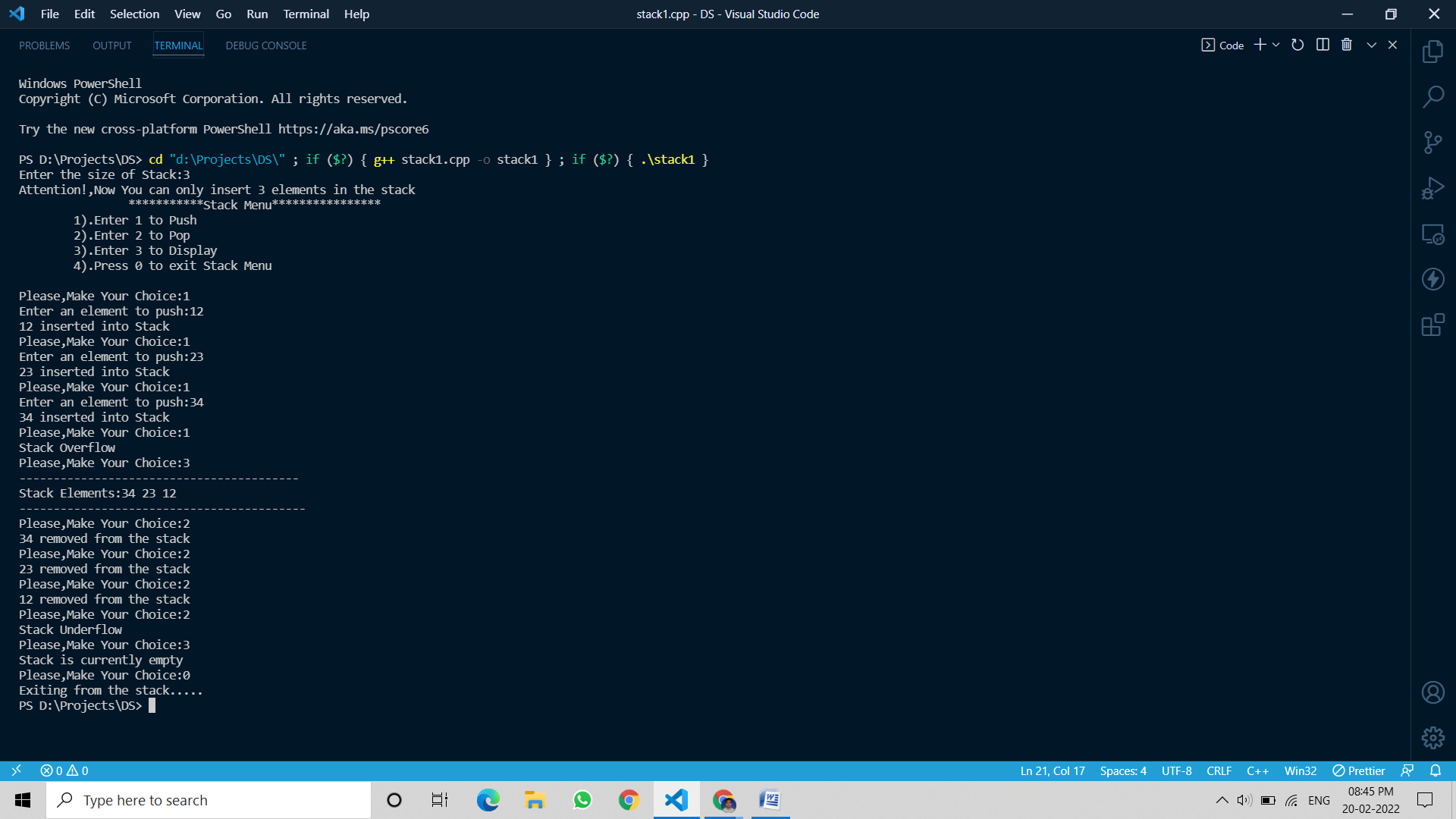
    }

*else*

*cout<<*"Stack is currently empty";

}

**Output:**

****

# Program 2. Write a program to evaluate a given postfix expression using stacks.

## Source Code:

*#include*<iostream>

*#include*<string.h>

*using* *namespace* std;

*#define* *n* 100

*int* *top=-*1,*stack*[*n*];

*void* *push*(*int* e)

{

*top++*;

*stack*[*top*]*=*e;

}

*float* *pop*()

{

*float* *x*;

*x=* *stack*[*top--*];

*return* *x*;

}

*int* *main*()

{

*char* *exp*[*n*];

*float* *a*,*b*,*c*;

*int* *i=*0;

*cout<<*"Enter the Postfix Expression:";

*cin>>exp*;

*while*(*i<=strlen*(*exp*))

    {

*if*(*exp*[*i*]*==*' ')

*continue*;

*else* *if*(*isdigit*(*exp*[*i*]))

        {

*push*(*exp*[*i*]*-*48);

        }

*else*

        {

*a=pop*();

*b=pop*();

*switch*(*exp*[*i*])

          {

*case* '+':

*c=b+a*;

*break*;

*case* '-':

*c=b-a*;

*break*;

*case* '\*':

*c=b\*a*;

*break*;

*case* '/':

*if*(*a==*0)

*cout<<*"Cannot divide by zero";

*else*

*c=b/a*;

*break*;

          }

*push*(*c*);

        }

*i++*;

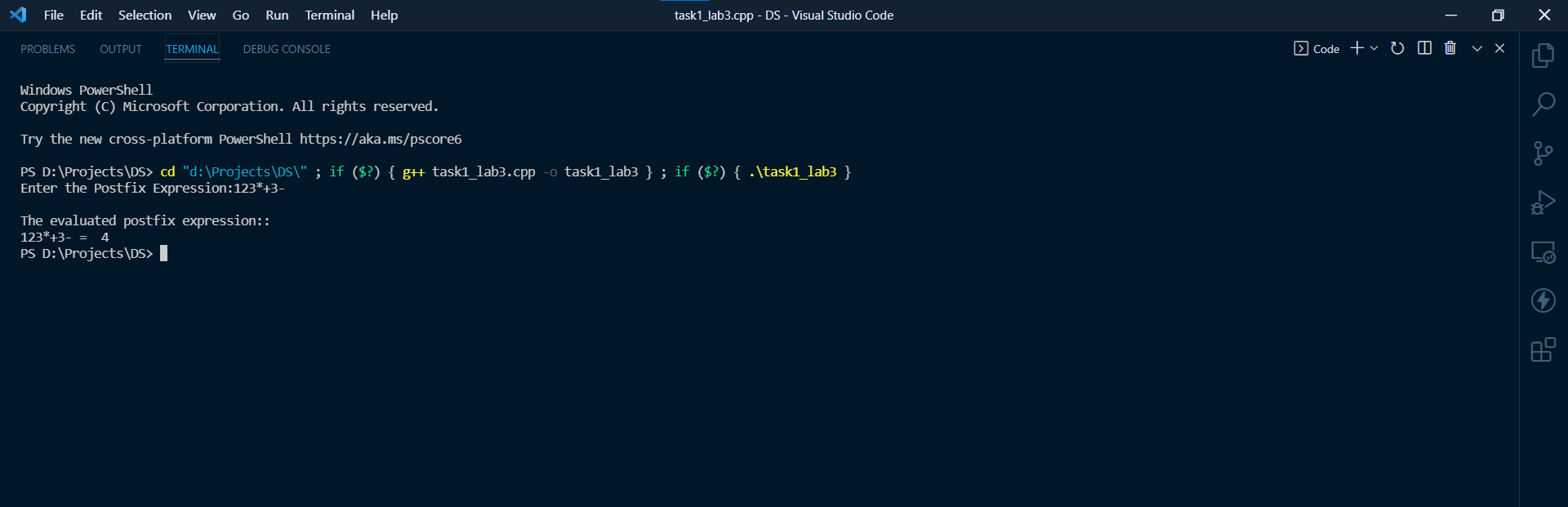
    }

*cout<<endl<<*"The evaluated postfix expression::"*<<endl*;

*cout<<exp<<*" =  "*<<stack*[*top*];

}

# Output:



# Program 3. Write a program to convert a given infix expression to postfix form using stacks.

## Source Code:

*#include* <iostream>

*#define* *n* 100

*using* *namespace* std;

*char* *stack*[*n*];

*int* *top* *=* *-*1;

*void* *push*(*char* c)

{

*stack*[*++top*] *=* c;

}

*char* *empty*()

{

*return* *stack*[*top--*];

}

*char* *pop*()

{

*char* *ch* *=* *stack*[*top*];

*top--*;

*return* *ch*;

}

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

{

*if* (symbol *==* '(')

*return* 0;

*if* (symbol *==* '+' *||* symbol *==* '-')

*return* 1;

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

*return* 2;

*return* 0;

}

*int* *main*()

{

*char* *exp*[100];

*char* *\*e*, *x*;

*cout* *<<* "Enter the infix Expression:";

*cin* *>>* *exp*;

*printf*("\n");

*e* *=* *exp*;

*cout* *<<* "The postfix expression of " *<<* *exp* *<<* " = ";

*while* (*\*e* *!=* '\0')

    {

*if* (*isalnum*(*\*e*))

        {

*cout* *<<* *\*e* *<<* " ";

        }

*else* *if* (*\*e* *==* '(')

*push*(*\*e*);

*else* *if* (*\*e* *==* ')')

        {

*while* ((*x* *=* *pop*()) *!=* '(')

*cout* *<<* *x* *<<* " ";

        }

*else*

        {

*while* (*precedence*(*stack*[*top*]) *>=* *precedence*(*\*e*))

*cout* *<<* *pop*() *<<* " ";

*push*(*\*e*);

        }

*e++*;

    }

*while* (*top* *!=* *-*1)

    {

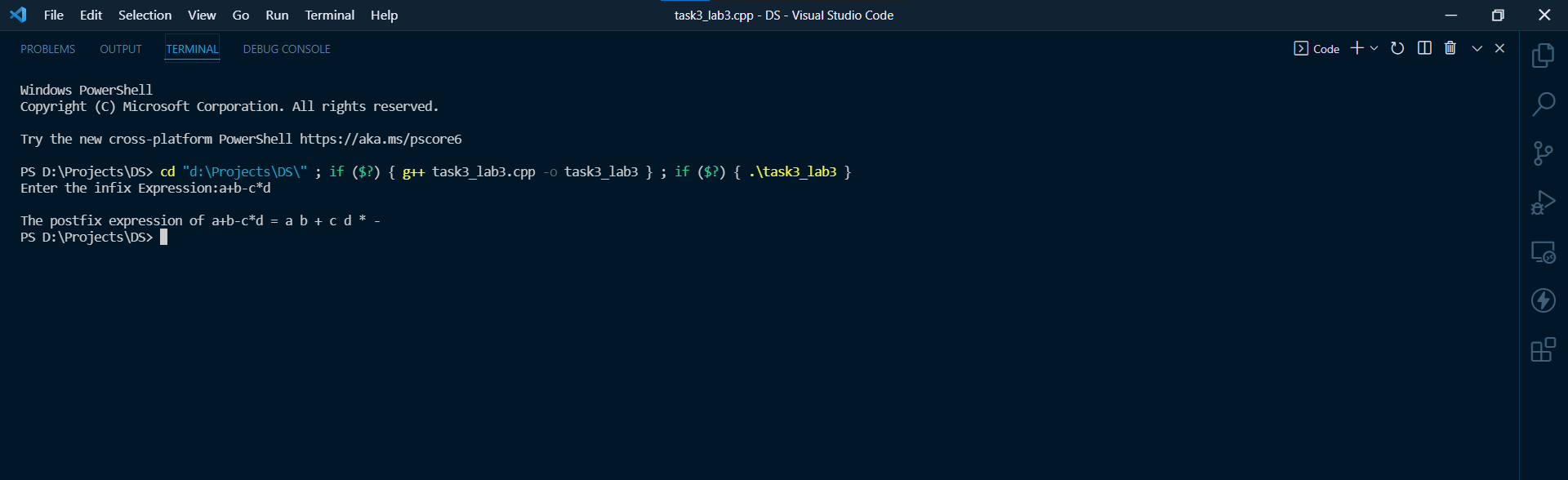
*cout* *<<* *pop*() *<<* " ";

    }

*return* 0;

}

## Output:



# Program 4. Write a program to implement circular queue using arrays

## Source Code:

*#include* <iostream>

*using* *namespace* std;

*int* *queue*[1000], *n*, *front* *=* *-*1, *rear* *=* *-*1;

*void* *insertion*();

*void* *deletion*();

*void* *display*();

*void* *top*();

*void* *last*();

*int* *main*()

{

*cout* *<<* "Enter Size of Queue:";

*cin* *>>* *n*;

*cout* *<<* "Attention! Size of Queue is " *<<* *n* *<<* " now" *<<* *endl*;

*int* *ch*;

*cout* *<<* "\n\n\t\t------------------------Circular Queue Menu---------------------";

*cout* *<<* "\n\t\t\t\t1)Enter 1 to Enqueue";

*cout* *<<* "\n\t\t\t\t2)Enter 2 to Dequeue";

*cout* *<<* "\n\t\t\t\t3)Enter 3 to Display the Queue";

*cout* *<<* "\n\t\t\t\t4)Enter 4 to Display Top";

*cout* *<<* "\n\t\t\t\t5)Enter 5 to Display Last";

*cout* *<<* "\n\t\t\t\t6)Enter 0 to Exit";

*while* (*ch* *!=* 0)

    {

*cout* *<<* "\nEnter Choice:";

*cin* *>>* *ch*;

*switch* (*ch*)

        {

*case* 1:

*insertion*();

*break*;

*case* 2:

*deletion*();

*break*;

*case* 3:

*display*();

*break*;

*case* 4:

*top*();

*break*;

*case* 5:

*last*();

*break*;

*case* 0:

        {

*cout* *<<* "Exited From Circular Queue menu";

*exit*(0);

        }

*break*;

*default*:

*cout* *<<* "Invalid Respnse";

        }

    }

*return* 0;

}

*void* *insertion*()

{

*int* *data*;

*if* ((*rear* *+* 1) *%* *n* *==* *front*)

    {

*cout* *<<* "Queue Overflow!";

    }

*else* *if* (*front* *==* *-*1 *&&* *rear* *==* *-*1)

    {

*front* *=* *rear* *=* 0;

*cout* *<<* "Enter an element to insert in Queue:";

*cin* *>>* *data*;

*queue*[*rear*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

*else*

    {

*cout* *<<* "Enter element:";

*cin* *>>* *data*;

*rear* *=* (*rear* *+* 1) *%* *n*;

*queue*[*rear*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

}

*void* *deletion*()

{

*if* (*front* *==* *-*1)

    {

*cout* *<<* "Queue Underflow!";

    }

*else* *if* (*front* *==* *rear*)

    {

*cout* *<<* *queue*[*front*] *<<* " deleted from queue";

*front* *=* *rear* *=* *-*1;

    }

*else*

    {

*cout* *<<* *queue*[*front*] *<<* " deleted from queue";

*front* *=* (*front* *+* 1) *%* *n*;

    }

}

*void* *top*()

{

*cout<<*"Top element of the Queue:"*<<queue*[*front*];

}

*void* *last*()

{

*cout<<*"Last element of the Queue:"*<<queue*[*rear*];

}

*void* *display*()

{

*if* (*front* *==* *-*1 *&&* *rear* *==* *-*1)

    {

*cout* *<<* "Queue is empty Currently!";

    }

*else*

    {

*cout* *<<* "-----------------------------------------------" *<<* *endl*;

*cout* *<<* "Queue Elements:";

*for* (*int* *i* *=* *front*; *i* *!=* *rear*; *i* *=* (*i* *+* 1) *%* *n*)

        {

*cout* *<<* *queue*[*i*] *<<* " ";

        }

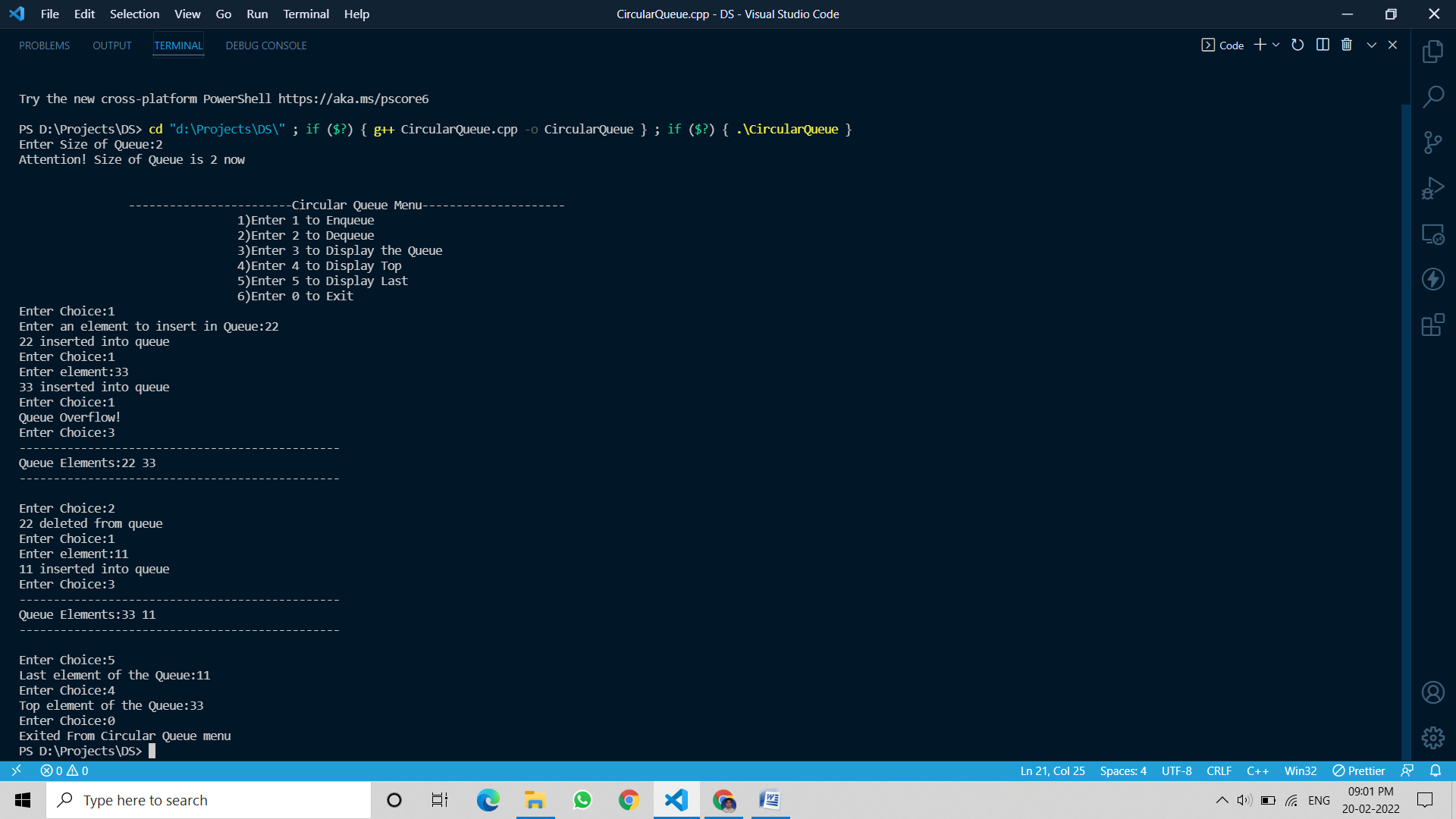
*cout* *<<* *queue*[*rear*];

*cout* *<<* *endl* *<<* "-----------------------------------------------" *<<* *endl*;

    }

}

## Output:



# Program5. Write a program to implement double ended queue (de queue) using arrays.

## Source Code:

*#include* <iostream>

*using* *namespace* std;

*int* *queue*[1000],*n* ,*front* *=* *-*1, *rear* *=* *-*1;

*void* *InsertAtFront*();

*void* *InsertAtRear*();

*void* *DeleteFromFront*();

*void* *DeleteFromRear*();

*void* *display*();

*int* *main*()

{

*cout<<*"Enter the size of Queue:";

*cin>>n*;

*int* *ch*;

*cout* *<<* "\n\n\t\t\t-------------Double Ended Queue Menu---------------";

*cout* *<<* "\n\t\t\t1).Insert element At the front side";

*cout* *<<* "\n\t\t\t2).Insert element At the rear side";

*cout* *<<* "\n\t\t\t3).Delete element from the front side";

*cout* *<<* "\n\t\t\t4).Delete element from the rear side";

*cout* *<<* "\n\t\t\t5).Display";

*cout* *<<* "\n\t\t\t6).Press 0 to Exit the Menu";

*while* (*ch* *!=* 0)

    {

*cout* *<<* "\nEnter Choice:";

*cin* *>>* *ch*;

*switch* (*ch*)

        {

*case* 1:

*InsertAtFront*();

*break*;

*case* 2:

*InsertAtRear*();

*break*;

*case* 3:

*DeleteFromFront*();

*break*;

*case* 4:

*DeleteFromRear*();

*break*;

*case* 5:

*display*();

*break*;

*case* 0:

        {

*cout* *<<* "Exited from Double Ended Queue menu";

*exit*(0);

        }

*break*;

*default*:

*cout* *<<* "Invalid response!";

        }

    }

*return* 0;

}

*void* *InsertAtFront*()

{

*int* *data*;

*if* ((*front* *==* 0 *&&* *rear* *==* *n* *-* 1) *||* (*front* *==* *rear* *+* 1))

    {

*cout* *<<* "Queue Overflow!";

    }

*else* *if* (*front* *==* *-*1 *&&* *rear* *==* *-*1)

    {

*front* *=* *rear* *=* 0;

*cout* *<<* "Enter data:";

*cin* *>>* *data*;

*queue*[*front*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

*else* *if* (*front* *==* 0)

    {

*front* *=* *n* *-* 1;

*cout* *<<*"Enter data:";

*cin* *>>* *data*;

*queue*[*front*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

*else*

    {

*front--*;

*cout* *<<*"Enter data:";

*cin* *>>* *data*;

*queue*[*front*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

}

*void* *InsertAtRear*()

{

*int* *data*;

*if* ((*front* *==* 0 *&&* *rear* *==* *n* *-* 1) *||* (*front* *==* *rear* *+* 1))

    {

*cout* *<<* "Queue Overflow!";

    }

*else* *if* (*front* *==* *-*1 *&&* *rear* *==* *-*1)

    {

*front* *=* *rear* *=* 0;

*cout* *<<* "Enter data:";

*cin* *>>* *data*;

*queue*[*rear*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

*else* *if* (*rear* *==* *n* *-* 1)

    {

*rear* *=* 0;

*cout* *<<* "Enter data:";

*cin* *>>* *data*;

*queue*[*rear*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

*else*

    {

*rear++*;

*cout* *<<* "Enter data:";

*cin* *>>* *data*;

*queue*[*rear*] *=* *data*;

*cout* *<<* *data* *<<* " inserted into queue";

    }

}

*void* *DeleteFromFront*()

{

*if* (*front* *==* *-*1 *&&* *rear* *==* *-*1)

    {

*cout* *<<* "Queue Undeflow!";

    }

*else* *if* (*front* *==* *rear*)

    {

*cout* *<<* *queue*[*front*] *<<* " deleted from queue";

*front* *=* *rear* *=* *-*1;

    }

*else* *if* (*front* *==* *n* *-* 1)

    {

*cout* *<<* *queue*[*front*] *<<* " deleted from queue";

*front* *=* 0;

    }

*else*

    {

*cout* *<<* *queue*[*front*] *<<* " deleted from queue";

*front++*;

    }

}

*void* *DeleteFromRear*()

{

*if* (*front* *==* *-*1 *&&* *rear* *==* *-*1)

    {

*cout* *<<* "Queue Undeflow!";

    }

*else* *if* (*front* *==* *rear*)

    {

*cout* *<<* *queue*[*rear*] *<<* " deleted from queue";

*front* *=* *rear* *=* *-*1;

    }

*else* *if* (*rear* *==* 0)

    {

*cout* *<<* *queue*[*rear*] *<<* " deleted from queue";

*rear* *=* *front* *-* 1;

    }

*else*

    {

*cout* *<<* *queue*[*rear*] *<<* " deleted from queue";

*rear--*;

    }

}

*void* *display*()

{

*if* (*front* *==* *-*1 *&&* *rear* *==* *-*1)

    {

*cout* *<<* "Queue Undeflow!";

    }

*else*

    {

*cout* *<<* "---------------------------------------" *<<* *endl*;

*cout* *<<* "Queue Elemenets:";

*for* (*int* *i* *=* *front*; *i* *!=* *rear*; *i* *=* (*i* *+* 1) *%* *n*)

        {

*cout* *<<* *queue*[*i*] *<<* " ";

        }

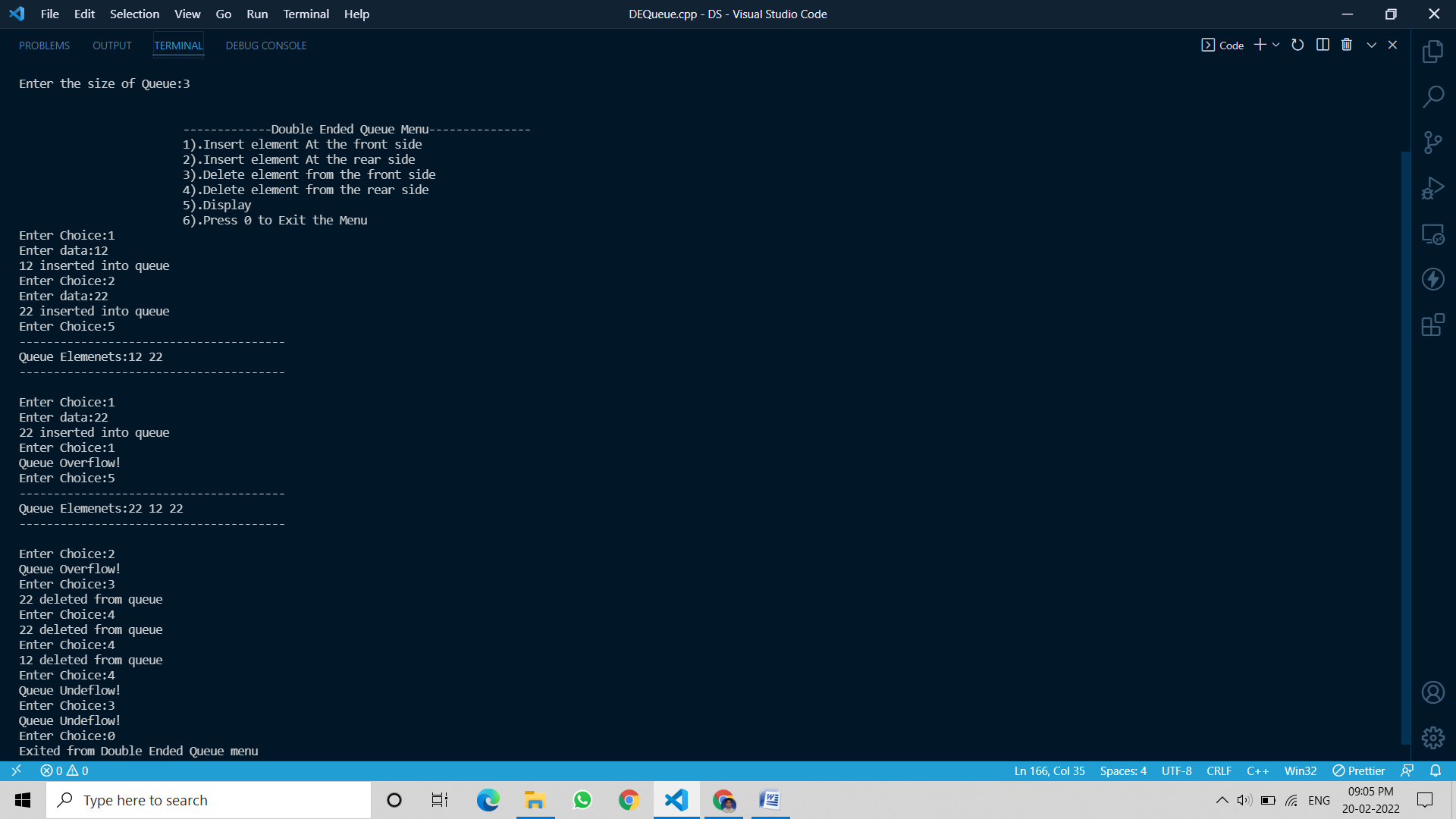
*cout* *<<* *queue*[*rear*];

*cout* *<<* *endl<<* "---------------------------------------" *<<* *endl*;

    }

}

## Output:



# Program6. Write a program to implement a stack using two queues such that the push operation runs in constant time and the pop operation runs in linear time.

## Source Code:

*#include* <bits/stdc++.h>

*using* *namespace* std;

*class* Stack

{

    queue*<int>* *q1*, *q2*;

*int* *curr\_size*;

*public:*

*void* *push*(*int* x)

    {

*q1*.*push*(x);

    }

*void* *pop*()

    {

*if* (*q1*.*empty*())

*return*;

*while* (*q1*.*size*() *!=* 1)

        {

*q2*.*push*(*q1*.*front*());

*q1*.*pop*();

        }

*q1*.*pop*();

        queue*<int>* *q* *=* *q1*;

*q1* *=* *q2*;

*q2* *=* *q*;

    }

*int* *top*()

    {

*if* (*q1*.*empty*())

*return* *-*1;

*while* (*q1*.*size*() *!=* 1)

        {

*q2*.*push*(*q1*.*front*());

*q1*.*pop*();

        }

*int* *temp* *=* *q1*.*front*();

*q1*.*pop*();

*q2*.*push*(*temp*);

        queue*<int>* *q* *=* *q1*;

*q1* *=* *q2*;

*q2* *=* *q*;

*return* *temp*;

    }

};

*int* *main*()

{

    Stack *s*;

*s*.*push*(33);

*cout* *<<* "Elements Puhed:" *<<* *endl*;

*s*.*push*(234);

*cout* *<<* "Elements Puhed:" *<<* *endl*;

*s*.*push*(123);

*cout* *<<* "Elements Puhed:" *<<* *endl*;

*s*.*push*(223);

*cout* *<<* "Elements Puhed:" *<<* *endl*;

*s*.*push*(323);

*cout* *<<* "Elements Puhed:" *<<* *endl*;

*s*.*push*(423);

*cout* *<<* "Elements Puhed:" *<<* *endl*;

*s*.*push*(23);

*cout* *<<* "Elements Puhed:" *<<* *endl*;

*cout* *<<* "Current Top elememnt:" *<<* *s*.*top*() *<<* *endl*;

*s*.*pop*();

*cout* *<<* "Elements Poped:" *<<* *endl*;

*cout* *<<* "Current Top elememnt:" *<<* *s*.*top*() *<<* *endl*;

*s*.*pop*();

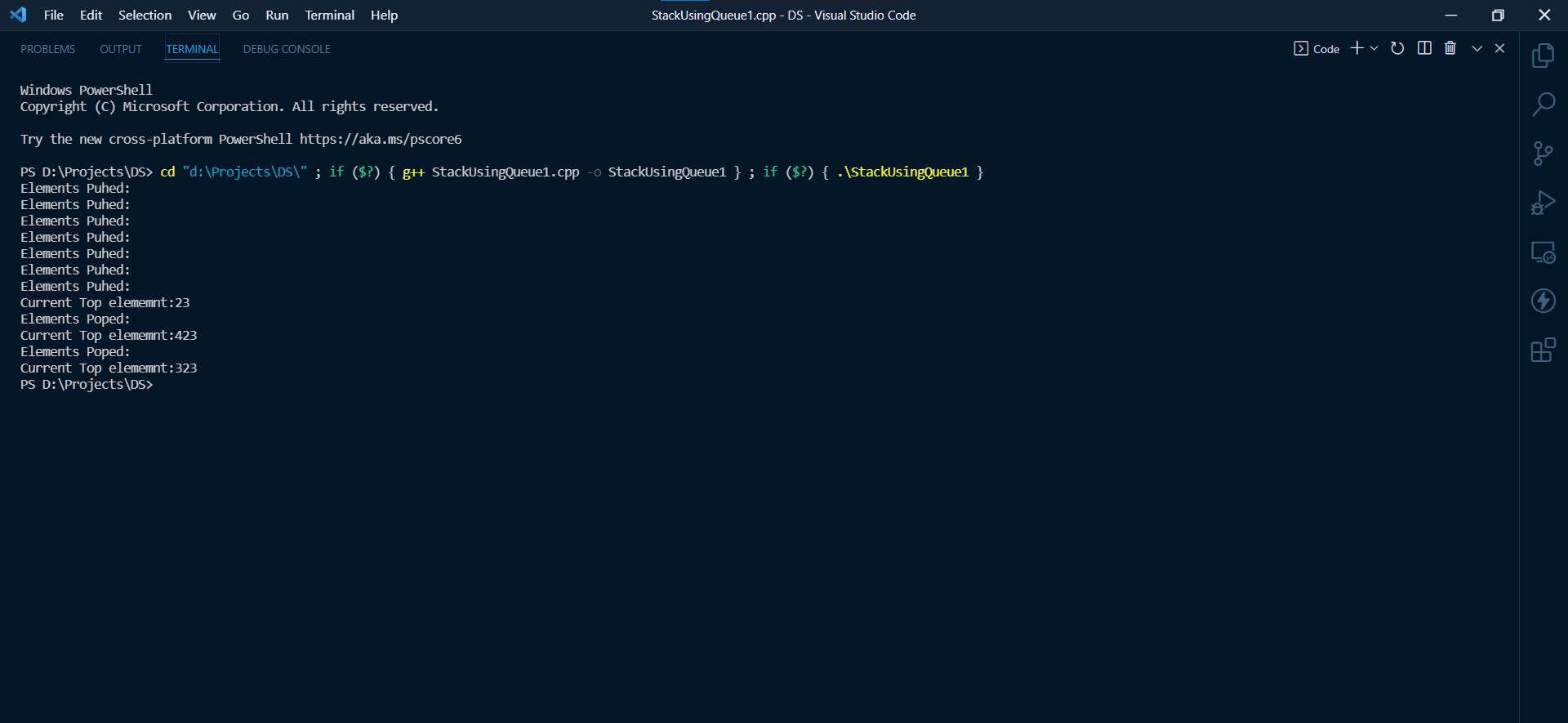
*cout* *<<* "Elements Poped:" *<<* *endl*;

*cout* *<<* "Current Top elememnt:" *<<* *s*.*top*() *<<* *endl*;

*return* 0;

}

## Output:



# Program7. Write a program to implement a stack using two queues such that the push operation runs in linear time and the pop operation runs in constant time.

## Source Code:

*#include* <bits/stdc++.h>

*using* *namespace* std;

*class* Stack

{

    queue*<int>* *q1*, *q2*;

*public:*

*void* *push*(*int* x)

    {

*q2*.*push*(x);

*while* (*!q1*.*empty*())

        {

*q2*.*push*(*q1*.*front*());

*q1*.*pop*();

        }

        queue*<int>* *q* *=* *q1*;

*q1* *=* *q2*;

*q2* *=* *q*;

    }

*void* *pop*()

    {

*if* (*q1*.*empty*())  *return* ;

*q1*.*pop*();

    }

*int* *top*()

    {

*if* (*q1*.*empty*())  *return* *-*1;

*return* *q1*.*front*();

    }

 };

*int* *main*()

{

    Stack *s*;

*s*.*push*(33);

*cout<<*"Elements Puhed:"*<<endl*;

*s*.*push*(234);

*cout<<*"Elements Puhed:"*<<endl*;

*s*.*push*(123);

*cout<<*"Elements Puhed:"*<<endl*;

*s*.*push*(223);

*cout<<*"Elements Puhed:"*<<endl*;

*s*.*push*(323);

*cout<<*"Elements Puhed:"*<<endl*;

*s*.*push*(423);

*cout<<*"Elements Puhed:"*<<endl*;

*s*.*push*(23);

*cout<<*"Elements Puhed:"*<<endl*;

*cout<<*"Current Top elememnt:"*<<s*.*top*()*<<endl*;

*s*.*pop*();

*cout<<*"Elements Poped:"*<<endl*;

*cout* *<<*"Current Top elememnt:"*<<s*.*top*()*<<endl*;

*s*.*pop*();

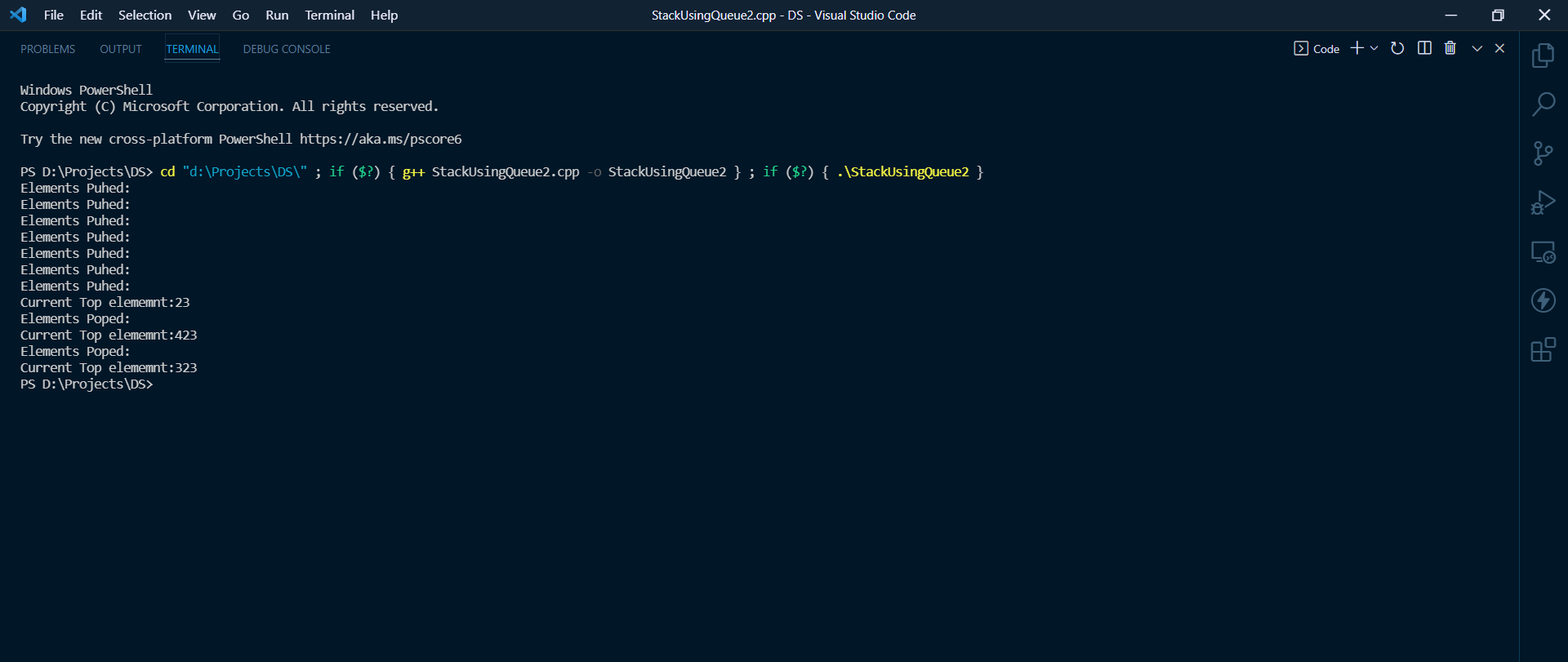
*cout<<*"Elements Poped:"*<<endl*;

*cout* *<<*"Current Top elememnt:"*<<s*.*top*()*<<endl*;

*return* 0;

}

## Output:



# Program8. Write a program to implement a queue using two stacks such that the enqueue operation runs in constant time and the dequeue operation runs in linear time.

## Source Code:

*#include* <iostream>

*using* *namespace* std;

*int* *stk1*[1000], *stk2*[1000], *n*, *top1* *=* *-*1, *top2* *=* *-*1, *count* *=* 0;

*void* *stack1\_push*(*int* a);

*int* *stack1\_pop*();

*void* *stack2\_push*(*int* a);

*int* *stack2\_pop*();

*void* *queue\_insert*();

*void* *queue\_delete*();

*void* *display*();

*int* *main*()

{

*cout* *<<* "Enter Size of Queue:";

*cin* *>>* *n*;

*cout<<*"!Warning! The size of the Queue is now "*<<n<<*" Only"*<<endl*;

*cout* *<<* "\n\n\t\t\t-------------------Queue Using Stack Menu-------------------------";

*cout* *<<* "\n\t\t\t\t1).Insert element in the Queue";

*cout* *<<* "\n\t\t\t\t2).Delete element from the Queue";

*cout* *<<* "\n\t\t\t\t3).Display element in the Queue";

*cout* *<<* "\n\t\t\t\t4).Enter 0 to Exit the Menu";

*int* *ch*;

*while* (*ch* *!=* 0)

    {

*cout* *<<*"\nChoice:";

*cin* *>>* *ch*;

*switch* (*ch*)

        {

*case* 1:

*queue\_insert*();

*break*;

*case* 2:

*queue\_delete*();

*break*;

*case* 3:

*display*();

*break*;

*case* 0:

        {

*cout<<*"Exited From Queue Menu....";

*exit*(0);

        }

*break*;

*default*:

        {

*cout* *<<* "Invalid";

        }

        }

    }

*return* 0;

}

*void* *stack1\_push*(*int* data)

{

*if* (*top1* *==* *n* *-* 1)

*cout* *<<* "!Overflow";

*else*

    {

*top1++*;

*stk1*[*top1*] *=* data;

    }

}

*int* *stack1\_pop*()

{

*if* (*top1* *==* *-*1)

*cout* *<<* "underflow!";

*else*

    {

*int* *p* *=* *stk1*[*top1*];

*top1--*;

*return* *p*;

    }

}

*void* *stack2\_push*(*int* data)

{

*if* (*top2* *==* *n* *-* 1)

*cout* *<<* "Overflow!";

*else*

    {

*top2++*;

*stk2*[*top2*] *=* data;

    }

}

*int* *stack2\_pop*()

{

*if* (*top2* *==* *-*1)

*cout* *<<* "underflow!";

*else*

    {

*int* *p* *=* *stk2*[*top2*];

*top2--*;

*return* *p*;

    }

}

*void* *queue\_insert*()

{

*if* (*top1* *==* *n* *-* 1)

*cout* *<<* "!Queue Overflow";

*else*{

*int* *e*;

*cout* *<<* "Enter an element:";

*cin* *>>* *e*;

*cout* *<<* *e* *<<* " inserted into Queue";

*stack1\_push*(*e*);

*count++*;

        }

}

*void* *queue\_delete*()

{

*if* (*top1* *==* *-*1 *&&* *top2* *==* *-*1)

*cout* *<<* "Queue Underflow!";

*else*

    {

*for* (*int* *i* *=* 0; *i* *<* *count*; *i++*)

        {

*int* *x* *=* *stack1\_pop*();

*stack2\_push*(*x*);

        }

*int* *b* *=* *stack2\_pop*();

*cout* *<<* *b* *<<* " deleted from queue";

*cout* *<<* *endl*;

*count--*;

*for* (*int* *i* *=* 0; *i* *<* *count*; *i++*)

        {

*int* *y* *=* *stack2\_pop*();

*stack1\_push*(*y*);

        }

    }

}

*void* *display*()

{

*if* (*top1* *==* *-*1 *&&* *top2* *==* *-*1)

*cout* *<<* "Queue is empty currently";

*else*

    {

*cout* *<<* "------------------------------------------" *<<* *endl*;

*cout* *<<* "Queue Elements:";

*for* (*int* *i* *=* 0; *i* *<=* *top1*; *i++*)

        {

*cout* *<<* *stk1*[*i*] *<<* " ";

        }

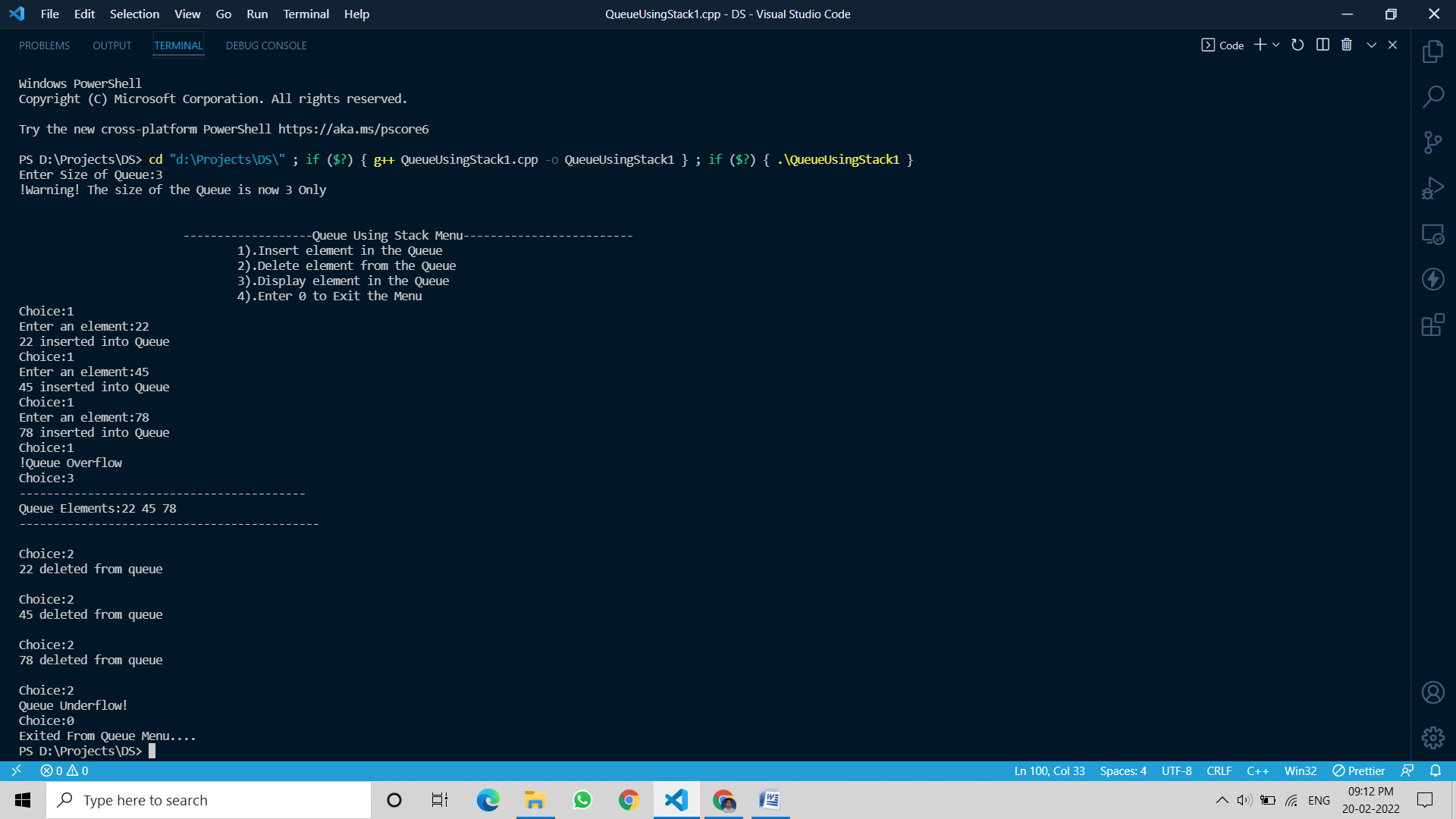
*cout* *<<* *endl*

*<<* "--------------------------------------------" *<<* *endl*;

    }

}

## Output:



# Program9. Write a program to implement a queue using two stacks such that the enqueue operation runs in linear time and the dequeue operation runs in constant time.

## Source Code:

*#include* <iostream>

*using* *namespace* std;

*int* *stk1*[1000], *stk2*[1000],*n*;

*int* *top1* *=* *-*1, *top2* *=* *-*1;

*void* *stack1\_push*(*int* data);

*int* *stack1\_pop*();

*void* *stack2\_push*(*int* x);

*int* *stack2\_pop*();

*void* *QueueInsert*();

*int* *QueueDelete*();

*void* *display*();

*int* *main*()

{

*cout* *<<* "Enter Size of Queue:";

*cin* *>>* *n*;

*cout<<*"!Warning! The size of the Queue is now "*<<n<<*" Only"*<<endl*;

*cout* *<<* "\n\n\t\t\t-------------------Queue Using Stack Menu-------------------------";

*cout* *<<* "\n\t\t\t\t1).Insert element in the Queue";

*cout* *<<* "\n\t\t\t\t2).Delete element from the Queue";

*cout* *<<* "\n\t\t\t\t3).Display element in the Queue";

*cout* *<<* "\n\t\t\t\t4).Enter 0 to Exit the Menu";

*int* *ch*;

*while* (*ch* *!=* 0)

    {

*cout* *<<* *endl*

*<<* "Choice:";

*cin* *>>* *ch*;

*switch* (*ch*)

        {

*case* 1:

        {

*QueueInsert*();

        }

*break*;

*case* 2:

*QueueDelete*();

*break*;

*case* 3:

*display*();

*break*;

*case* 0:

*exit*(0);

break;

*default*:

*cout* *<<* "Invalid";

        }

    }

*return* 0;

}

*void* *stack1\_push*(*int* data)

{

    {

*top1++*;

*stk1*[*top1*] *=* data;

    }

}

*int* *stack1\_pop*()

{

    {

*int* *a* *=* *stk1*[*top1*];

*top1--*;

*return* *a*;

    }

}

*void* *stack2\_push*(*int* x)

{

    {

*top2++*;

*stk2*[*top2*] *=* x;

    }

}

*int* *stack2\_pop*()

{

*int* *p* *=* *stk2*[*top2*];

*top2--*;

*return* *p*;

}

*void* *QueueInsert*()

{

*if* (*top1* *==* *n* *-* 1)

*cout* *<<* "!Queue Overflow";

*else*{

*while* (*top1* *!=* *-*1)

    {

*stack2\_push*(*stack1\_pop*());

    }

*int* *e*;

*cout<<*"Enter an element:";

*cin>>e*;

*cout<<e<<*" inserted into Queue";

*stack1\_push*(*e*);

*while* (*top2* *!=* *-*1)

    {

*stack1\_push*(*stack2\_pop*());

    }

        }

}

*int* *QueueDelete*()

{

*if* (*top1* *==* *-*1 *&&* *top2* *==* *-*1)

*cout* *<<* "Queue Underflow!";

*else*

    {

*cout<<stk1*[*top1*]*<<*" deleted from Queue";

*int* *d* *=* *stk1*[*top1*];

*top1--*;

*return* *d*;

    }

}

*void* *display*()

{

*if* (*top1* *==* *-*1 *&&* *top2* *==* *-*1)

    {

*cout* *<<* "Queue is empty now!";

    }

*else*

    {

*cout<<*"------------------------------------------------------"*<<endl*;

*cout<<*"Queue Elements:";

*for* (*int* *i* *=* *top1*; *i* *>=* 0; *i--*)

        {

*cout* *<<* *stk1*[*i*] *<<* " ";

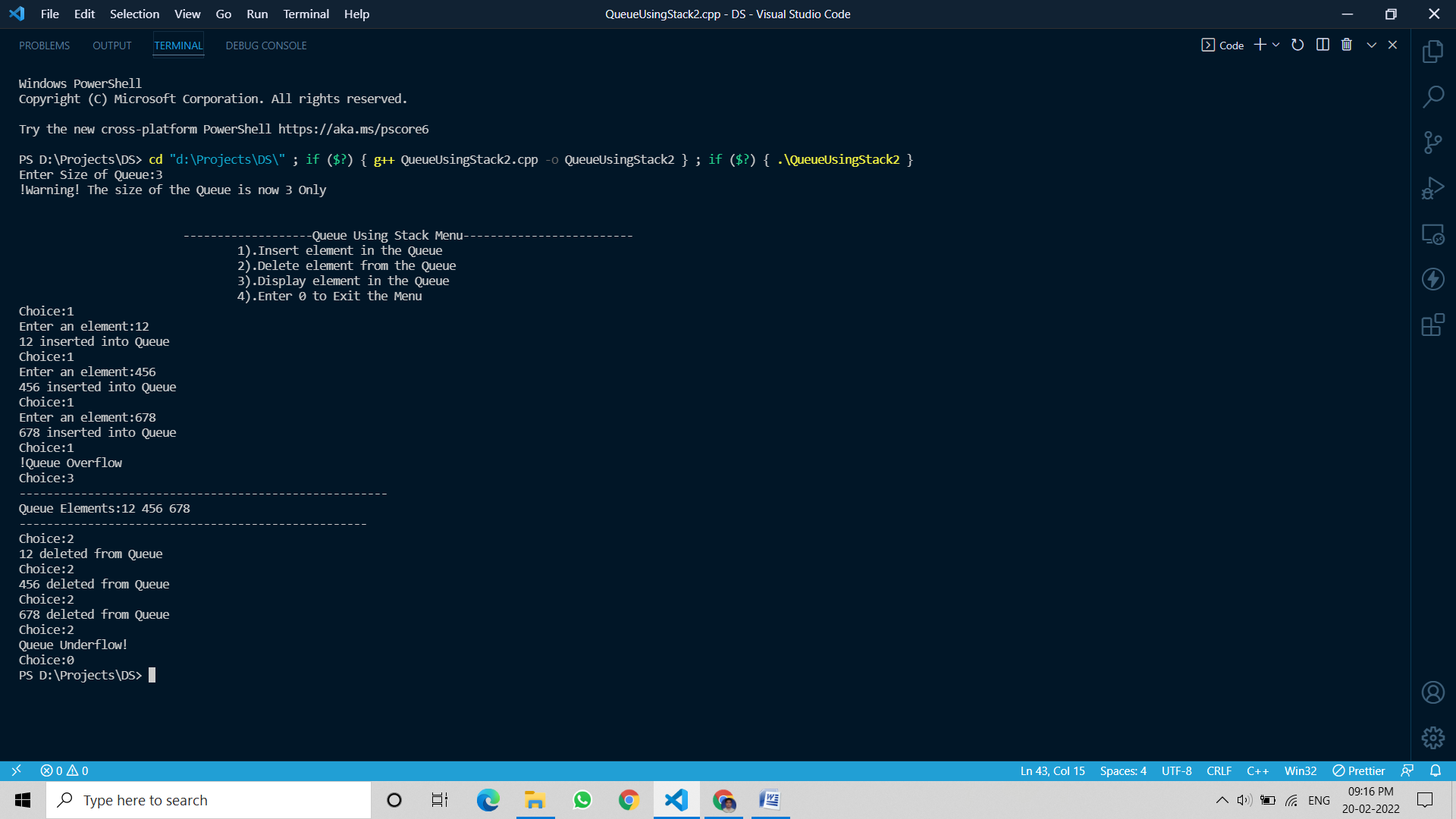
        }

*cout<<endl<<*"---------------------------------------------------";

    }

}

## Output:



# Program10. Write program to implement the Single linked list

## Source Code:

## Output:

*#include* <iostream>

*using* *namespace* std;

*struct* node

{

*int* *e*;

*struct* node *\*next*;

};

*struct* node *\*head* *=* *NULL*, *\*ptr*;

*void* *insertAtBegin*()

{

*struct* node *\*temp*;

*int* *data*;

*cout<<endl<<*"Enter data:";

*cin* *>>* *data*;

*temp* *=* *new* node;

*temp*->*e* *=* *data*;

*temp*->*next* *=* *NULL*;

*if* (*head* *==* *NULL*)

    {

*head* *=* *temp*;

*cout* *<<* *data* *<<* " inserted into begining of linked list";

    }

*else*

    {

*temp*->*next* *=* *head*;

*head* *=* *temp*;

*cout* *<<* *data* *<<* " inserted into begining of linked list";

    }

}

*void* *insertAtBottom*()

{

*struct* node *\*temp*;

*int* *data*;

*cout<<endl<<* "Enter data:";

*cin* *>>* *data*;

*temp* *=* *new* node;

*temp*->*e* *=* *data*;

*temp*->*next* *=* *NULL*;

*if* (*head* *==* *NULL*)

    {

*head=temp*;

*cout* *<<* *data* *<<* " inserted into bottom of linked list";

    }

*else*

    {

*ptr=head*;

*while* (*ptr*->*next* *!=* *NULL*)

        {

*ptr* *=* *ptr*->*next*;

        }

*ptr*->*next* *=* *temp*;

*cout* *<<* *data* *<<* " inserted into bottom of  linked list";

    }

}

*void* *insertAtRandom*()

{

*struct* node *\*temp*;

*temp* *=* *new* node;

*int* *data*,*l*;

*cout<<*"Enter Location:";

*cin>>l*;

*cout* *<<* *endl<<* "Enter data:";

*cin* *>>* *data*;

*temp*->*e* *=* *data*;

*if*(*head==NULL*)

    {

*head=temp*;

*cout<<data<<*" inserted just after element "*<<l<<*" in the Linked List";

    }

*else*

    {

*ptr=head*;

*while*(*ptr*->*e!=l*)

        {

*ptr=ptr*->*next*;

        }

*temp*->*next=ptr*->*next*;

*ptr*->*next=temp*;

*cout<<data<<*" inserted just after element "*<<l<<*" in the Linked List";

    }

}

*void* *deleteFromBegin*()

{

*if* (*head* *==* *NULL*)

    {

*cout* *<<* "Underflow!No elements there";

    }

*else*

    {

*cout* *<<*"Element deleted from begining side in the linked list";

*ptr* *=* *head*;

*head* *=* *head*->*next*;

*ptr*->*next* *=* *NULL*;

*delete* (*ptr*);

    }

}

*void* *deleteFromBottom*()

{

*if* (*head* *==* *NULL*)

    {

*cout* *<<*"Underflow!No elements there";

    }

*else*

    {

*cout<<*"Element deleted from bottom side of the linked list";

*struct* node *\*prev*;

*ptr=head*;

*while* (*ptr*->*next* *!=* *NULL*)

        {

*prev=ptr*;

*ptr=ptr*->*next*;

        }

*prev*->*next* *=* *NULL*;

*delete* (*ptr*);

    }

}

*void* *deleteFromRandom*()

{

*int* *data*;

*cout<<*"Enter existing element to delete:";

*cin>>data*;

*if* (*head* *==* *NULL*)

    {

*cout* *<<* "Underflow!No elements there";

    }

*else*{

*struct* node *\*p*;

*cout<<*"Element deleted from selected place";

*ptr=head*;

*while*(*ptr*->*e!=data*)

    {

*p=ptr*;

*ptr=ptr*->*next*;

    }

*p*->*next=ptr*->*next*;

*delete*(*ptr*);

    }

}

*void* *display*()

{

*if* (*head* *==* *NULL*)

    {

*cout* *<<* "Linked List is Empty currently!";

    }

*else*

    {

*ptr* *=* *head*;

*cout<<endl<<* "-------------------------------------" *<<* *endl*;

*cout* *<<* "Linked List Elements:";

*while* (*ptr* *!=* *NULL*)

        {

*cout* *<<* *ptr*->*e* *<<* " ";

*ptr* *=* *ptr*->*next*;

        }

*cout* *<<* *endl<<* "----------------------------------------";

    }

}

*int* *main*()

{

*int* *ch*;

*cout* *<<* "\n\n\t\t-------------------Linked List Menu---------------------------";

*cout* *<<* "\n\t\t\t1).Insertion At The begining";

*cout* *<<* "\n\t\t\t2).Insertion At The bottom";

*cout* *<<* "\n\t\t\t3).Insertion At The Random";

*cout* *<<* "\n\t\t\t4).Deletion At The begining";

*cout* *<<* "\n\t\t\t5).Deletion At The bottom";

*cout* *<<* "\n\t\t\t6).Deletion At The Random";

*cout* *<<* "\n\t\t\t7).Display Linked List";

*cout* *<<* "\n\t\t\t8).Enter 0 to Exit the Menu";

*while* (*ch* *!=* 0)

    {

*cout* *<<* *endl*

*<<* "Enter choice:";

*cin* *>>* *ch*;

*switch* (*ch*)

        {

*case* 1:

*insertAtBegin*();

*break*;

*case* 2:

*insertAtBottom*();

*break*;

*case* 3:

*insertAtRandom*();

*break*;

*case* 4:

*deleteFromBegin*();

*break*;

*case* 5:

*deleteFromBottom*();

*break*;

*case* 6:

*deleteFromRandom*();

*break*;

*case* 7:

*display*();

*break*;

*case* 0:{

*cout<<*"Exited from Menu";

*exit*(0);

        }

*break*;

*default*:

*cout* *<<* "Invalid Response!";

        }

    }

}

## Output:

