Program :Write a c++ program to perform stack operation:

Date:30-OCT-2024

Autho[redacted]

```
#include<iostream>
using namespace std;
#define n 5
int arr[n];
int top=-1;

void push(){
   if(top>=n-1){
      cout<<"stack is full"<<endl;
   }else{
                int element;
                cout<<"enter a element\n";
      cin>>element;
      arr[++top]=element;
      cout<<element<<"pushed into stack";
   }
}
void pop(){
   if(top==-1){
      cout<<"stack is empty"<<endl;
   }else{
                cout<<"element deleted from stack is"<<arr[top--];
   }
}
void display(){
```

```
    if(top==-1){
        cout<<"stack is empty";
    }else{
        cout<<"stack elements are:\n";
        for(int i=0;i<=top;i++){
            cout<<arr[i]<<" ";
        }
                cout<<endl;
    }
}
int main(){
        int ch;
        cout<<"\n1.push\n2.pop\n3.display\n4.exit\n";
        while(true){
                cout<<"enter your choice\n";
                cin>>ch;
                switch(ch){
                        case 1:
                                push();
                                break;
                        case 2:
                                pop();
                                break;
                        case 3:
                                display();
                                break;
                                                        case 4:
                                return 0;
                                break;
                        default:
```

```
                cout<<"invalid";
        }
    }
}
```

```
[24251420@localhost ~]$ ./a.out

1.push
2.pop
3.display
4.exit

enter your choice1

enter a element10
10
pushed into stack
enter your choice1

enter a element20
20
pushed into stack
enter your choice3

stack elements are:10 20

enter your choice2

element deleted from stack is20
enter your choice3

stack elements are:10
```

Program : Write a c++ program to perform queue operation:

Date:30-OCT-2024

Author:

```
#include<iostream>
using namespace std;
#define max 5
int q[max];
int last=-1;
void enqueue(){
    if(last==max-1){
        cout<<"queue is full\n";
    }else{
        int element;
        cout<<"enter a element\n";
        cin>>element;
        q[++last]=element;
        cout<<"element inserted\n";
    }
}
void dequeue()
{
    if(last==-1){
        cout<<"queue is empty\n";
    }else{
        cout<<"element deleted is:"<<q[0]<<"\n";
        for(int i=0;i<last;i++){
        q[i]=q[i+1];
        }
```

```
        last--;
    }
}
void display(){
    if(last==-1){
        cout<<"queueis empty\n";
    }else{
        cout<<"elements in queue are"<<endl;
        for(int i=0;i<=last;i++){
            cout<<q[i]<<" ";
        }
        cout<<"\n";

    }

}
int main(){
    int ch;
    cout<<"\n1.insert\n2.delete\n3.display\n4.exit\n";
    while(true){
        cout<<"enter your choice\n";
        cin>>ch;
        switch(ch){
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
```

```
                display();
                break;
                                        case 4:
                return 0;
                break;
            default:
                cout<<"invalid";
        }
    }
}
```

```
[24251420@localhost ~]$ ./a.out

1.insert
2.delete
3.display
4.exit
enter your choice
1
enter a element
10
element inserted
enter your choice
1
enter a element
20
element inserted
enter your choice
1
enter a element
30
element inserted
enter your choice
2
element deleted is:10
enter your choice
3
elements in queue are
20 30
```

Program 6:Write a c++ program to perform postfix evaluation:

Date:30-OCT-2024

Aut[redacted]

```
#include<iostream>
using namespace std;
int evaluatePostfix(const char* expression){
        int opstack[100];
        int top=-1;
        for(int i=0;expression[i]!='\0';++i)
        {
        char symb=expression[i];
        if(symb==' ') continue;

                if(symb>='0'&&symb<='9'){
                        opstack[++top]=symb-'0';
                }else{
                        int opnd1=opstack[top--];
                        int opnd2=opstack[top--];
                        switch(symb){
                                case '+':
                                        opstack[++top]=opnd1+opnd2;
                                        break;
                                case '-':
                                        opstack[++top]=opnd1-opnd2;
                                        break;
                                case '*':
                                        opstack[++top]=opnd1*opnd2;
                                        break;
```

```
                case '/':
                        opstack[++top]=opnd1/opnd2;
                        break;
                default:
                        cout<<"invalid operator:"<<symb<<endl;
                        return 0;
                }
        }
    }
    return opstack[top];
}
int main(){
    char expression[100];
    cout<<"enter a postfix expression";
    cin.getline(expression,100);
    cout<<"result:{"<<evaluatePostfix(expression)<<endl;
    return 0;
    }
```

```
[24251420@localhost ~]$ ./a.out
enter a postfix expression123+-
result:4
```

Program 5:Write a c++ program to perform prefix evaluation:

Date:30-OCT-2024

Auth[redacted]

```
#include<iostream>
#include<cstring>
using namespace std;
int evaluatePrefix(const char* expression){
    int opstack[100];
    int top=-1;
    for(int i=strlen(expression)-1;i>=0;--i)
    {
    char symb=expression[i];
    if(symb==' ') continue;

        if(symb>='0'&&symb<='9'){
            opstack[++top]=symb-'0';
        }else{
            int opnd1=opstack[top--];
            int opnd2=opstack[top--];
            switch(symb){
                case '+':
                    opstack[++top]=opnd1+opnd2;
                    break;
                case '-':
                    opstack[++top]=opnd1-opnd2;
                    break;
                case '*':
                    opstack[++top]=opnd1*opnd2;
```

```
                    break;
                case '/':
                    opstack[++top]=opnd1/opnd2;
                    break;
                default:
                    cout<<"invalid operator:"<<symb<<endl;
                    return 0;
                }
            }
        }
        return opstack[top];
}
int main(){
        char expression[100];
        cout<<"enter a prefix expression";
        cin.getline(expression,100);
        cout<<"result:{"<<evaluatePrefix(expression)<<endl;
        return 0;
        }
```

```
[24251420@localhost ~]$ ./a.out
enter a prefix expression+-123
result:2
```

Program 3:Write a c++ program to perform circular queue operation:

Date:30-OCT-2024

Auth[redacted]

```
#include<iostream>
using namespace std;
#define max 5
int q[max];
int front=-1,rear=-1;
void enqueue(){
    if((front==(rear+1)%max)){
        cout<<"queue is full\n";
    }else{
        int element;
        cout<<"enter a element\n";
        cin>>element;
        if(front==-1){
            front=0;
        }
        rear=(rear+1)%max;
        q[rear]=element;
        cout<<"element inserted\n";
    }
}
void dequeue()
{
    if(front==-1){
        cout<<"queue is empty\n";
    }else{
```

```
        cout<<"element deleted is:"<<q[front]<<"\n";
        if(front==rear){
            front=rear=-1;
        }else{
                                        front=(front+1)%max;
        }
    }
}
void display(){
    if(front==-1){
        cout<<"queue is empty\n";
    }else{
        cout<<"elements in circular queue are"<<endl;
        int i=front;
        while(true){
            cout<<q[i]<<" ";
            if(i==rear)
                break;
            i=(i+1)%max;
        }
        cout<<"\n";

    }
}
int main(){
    int ch;
    cout<<"\n1.insert\n2.delete\n3.display\n4.exit\n";
    while(true){
        cout<<"enter your choice\n";
        cin>>ch;
```

```
    switch(ch){
        case 1:
            enqueue();
            break;
        case 2:
            dequeue();
            break;
        case 3:
            display();
            break;
                                    case 4:
            return 0;
            break;
        default:
            cout<<"invalid";
    }
}
```

```
[24251420@localhost ~]$ ./a.ou

1.insert
2.delete
3.display
4.exit
enter your choice
1
enter a element
10
element inserted
enter your choice
1
enter a element
20
element inserted
enter your choice
1
enter a element
30
element inserted
enter your choice
2
element deleted is:10
enter your choice
3
elements in circular queue are
20 30
```

Program 8: Write a c++ program to Converting an Expression from Infix to Postfix with and without parenthesis.

Date:29-OCT-2024

Author [redacted]

```
#include <iostream>
#include <cstring>
using namespace std;
int precedence(char op) {
  if (op == '+' || op == '-') return 1;
  if (op == '*' || op == '/') return 2;
  return 0;
}
bool isOperator(char ch) {
  return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}
string infixToPostfix(const char* infix) {
  char opstk[100];
  int top = -1;
  string postfix = "";
  for (int i = 0; i < strlen(infix); ++i) {
    char symb = infix[i];

    if (isalnum(symb)) {
      postfix += symb;
    } else if (symb == '(') {
      opstk[++top] = symb;
    } else if (symb == ')') {
      while (top != -1 && opstk[top] != '(') {
```

```
            postfix += opstk[top--];
        }
        top--; // Pop the '('
    } else if (isOperator(symb)) {
        while (top != -1 && precedence(opstk[top]) >= precedence(symb) && opstk[top] != '(') {
            postfix += opstk[top--];
        }
        opstk[++top] = symb;
    }
  }
  while (top != -1) {
    postfix += opstk[top--];
  }
  return postfix;
}
int main() {
  char infix[100];
  cout << "Enter an infix expression with or without parentheses: ";
  cin.getline(infix, 100);
  string postfix = infixToPostfix(infix);
  cout << "Postfix expression: " << postfix << endl;
return 0;
}
```

OUTPUT:

```
Enter an infix expression with or without parentheses: a+b*c
Postfix expression: abc*+
```

```
Enter an infix expression with or without parentheses: q+e*(e/e)
Postfix expression: qeee/*+
```

Program :Write a c++ program to perform double ended queue operation:

Date:30-OCT-2024

Autho[redacted]

```
#include<iostream>
using namespace std;
#define max 5
int q[max];
int front=-1,rear=-1;
void insert_front(){
    if(front==0){
        cout<<"queue is full  at the front\n";
    }else{
        int element;
        cout<<"enter a element\n";
        cin>>element;
        if(front==-1){
            front=rear=0;
        }else{
            front=front+1;
        }
        q[front]=element;
        cout<<"element inserted\n";
    }
}
void insert_rear(){
    if(rear==max-1){
        cout<<"queue is full  at the rear\n";
    }else{
```

```
            int element;
            cout<<"enter a element\n";
            cin>>element;
            if(front==-1){
                front=rear=0;
            }
            rear=rear+1;
            q[rear]=element;
            cout<<"element inserted\n";
        }
    }
    void delete_front()
    {
        if(front==-1){
            cout<<"queue is empty at the front\n";
        }else{
            cout<<"element deleted from front:"<<q[front]<<"\n";
            front=front+1;
            if(front>rear){
                front=rear=-1;
            }
        }
    }
    void delete_rear()
    {
        if(rear==-1){
            cout<<"queue is empty at the rear\n";
        }else{
            cout<<"element deleted from rear:"<<q[rear]<<"\n";
            rear=rear-1;
```

```
        if(front<rear){
            front=rear=-1;
        }
    }
}
void display(){
    if(front==-1){
        cout<<"queue is empty\n";
    }else{
        cout<<"elements in circular queue are"<<endl;
        for(int i=front;i<rear;i++){
            cout<<q[i]<<" ";
        }
        cout<<"\n";

    }
}
int main(){
    int ch;
    cout<<"\n1.insert from front\n2insert from rear\n3.delete from front\n4.delete from
rear\n5.display\n6exit\n";
    while(true){
        cout<<"enter your choice\n";
        cin>>ch;
        switch(ch){
            case 1:
                insert_front();
                break;
            case 2:
                insert_rear();
                break;
```

```
            case 3:
                delete_front();
                break;
            case 4:
                delete_rear();
                break;
            case 5:
                display();
                break;
            case 6:
                return 0;
                break;
            default:
                cout<<"invalid";
        }
    }
}
```

```
^[[A[24251420@localhost ~]$ ./a.out

1.insert from front
2.insert from rear
3.delete from front
4.delete from rear
5.display
6.exit
enter your choice
1
enter a element
10
element inserted
enter your choice
2
enter a element
20
element inserted
enter your choice
5

elements in circular queue are
10 20
enter your choice
3

element deleted from front:10
enter your choice
4

element deleted from rear:20
```