

Data Structure Assignment - 3

Problems

- Write a C program to perform push and pop operation on a stack array
- Write a C program to perform push and pop operation on a linked stack
- Write a C program to reverse a string using stack
- Write a C program to evaluate postfix notation
- Write a C program to infix notation into equivalent postfix notation
- Write a C program to implement a circular queue
- Write a C program to implement input, output on restricted dequeues

Problem - 1

C Code

```
1 /**
2  * Data Structures Assignment - 3
3  * CS 392
4  * Problem 1:
5  *      Write a C program to perform push and pop operation on a stack array
6  *
7  * Joydeep Mukherjee
8  * CSE, 3rd Sem, 11000116030
9  * GCETTS 2017
10 *
11 **/
12
13 #include <stdio.h>
14 #include <stdlib.h>
15
16 int *stack;
17 int pos;
18 int size;
19
20
21 void init() {
22     // Prompt the user to enter the size of stack
23     printf("Enter Size of stack: ");
24     scanf("%d",&size);
25     stack = malloc(sizeof(int)*size);
26     pos = 0;
27 }
28 void push(int elem) {
29     if(size == pos) {
30         printf("Stack Overflow, %d not added\n",elem);
31         return;
32     }
33     stack[pos++] = elem;
34 }
35
36 int pop() {
37     if(pos == 0) {
```

```
38     printf("Stack is Empty\n");
39     return 0;
40 }
41 return stack[--pos];
42 }
43
44
45 int main(void ){
46     int temp;
47
48     init();
49
50     printf("Enter 0 to pop, -1 to exit\n");
51     do{
52         printf(">>> ");
53         scanf("%d", &temp);
54
55         if(temp)
56             push(temp);
57         else
58             if(temp==pop())
59                 printf("Popped: %d\n",temp);
60
61     } while(temp != -1);
62
63     return 0;
64 }
```

Output

```
1  → ./3_1
2  Enter Size of stack: 5
3  Enter 0 to pop, -1 to exit
4  >>> 5
5  >>> 9
6  >>> 4
7  >>> 3
8  >>> 4
9  >>> 8
10 Stack Overflow, 8 not added
11 >>> 0
12 Popped: 4
13 >>> 0
14 Popped: 3
15 >>> 0
16 Popped: 4
17 >>> 0
18 Popped: 9
19 >>> 0
20 Popped: 5
21 >>> 0
22 Stack is Empty
23 >>> 0
24 Stack is Empty
25 >>> 5
26 >>> 6
27 >>> 1
28 >>> 0
29 Popped: 1
```

```
30 >>> 0
31 Popped: 6
32 >>> 0
33 Popped: 5
34 >>> 0
35 Stack is Empty
36 >>> -1
37
```

Problem - 2

C Code

```
1 /**
2  * Data Structures Assignment - 3
3  * CS 392
4  * Problem 2:
5  *     Write a C program to perform push and pop operation on a linked stack
6  *
7  * Joydeep Mukherjee
8  * CSE, 3rd Sem, 11000116030
9  * GCETTS 2017
10 *
11 **/
12
13 #include <stdio.h>
14 #include <stdlib.h>
15
16
17 typedef struct node_t {
18     int val;
19     struct node_t *next;
20 }node;
21
22 node *stack;
23
24 void init() {
25     // Create Stack
26     stack = NULL;
27 }
28 void push(int elem) {
29     // Allocate New node
30     node *new = malloc(sizeof(node));
31     // Check for memory fault
32     if(!new) {
33         printf("Memory Fault\n");
34         return;
35     }
36     new->val = elem;
37     new->next = stack;
38
39     stack = new;
40 }
41 int pop() {
42     int temp;
43     // Check for empty stack
44     if(!stack) {
45         printf("Stack is Empty\n");
46         return 0;
47     }
```

```
47     }
48     temp = stack->val;
49     stack = stack->next;
50
51     return temp;
52 }
53
54
55 int main(void ){
56     int temp;
57
58     init();
59
60     printf("Enter 0 to pop, -1 to exit\n");
61     do{
62         printf(">>> ");
63         scanf("%d", &temp);
64
65         if(temp)
66             push(temp);
67         else
68             if(temp==pop())
69                 printf("Popped: %d\n",temp);
70
71     } while(temp != -1);
72
73     return 0;
74 }
75
```

Output

```
1  → ./3_2
2  Enter 0 to pop, -1 to exit
3  >>> 5
4  >>> 6
5  >>> 6
6  >>> 7
7  >>> 4
8  >>> 2
9  >>> 5
10 >>> 5
11 >>> 5
12 >>> 5
13 >>> 78
14 >>> 98
15 >>> 12
16 >>> 0
17 Popped: 12
18 >>> 0
19 Popped: 98
20 >>> 0
21 Popped: 78
22 >>> 0
23 Popped: 5
24 >>> 0
25 Popped: 5
26 >>> 0
27 Popped: 5
```

```
28 >>> 0
29 Popped: 5
30 >>> 0
31 Popped: 2
32 >>> 0
33 Popped: 4
34 >>> 0
35 Popped: 7
36 >>> 0
37 Popped: 6
38 >>> 0
39 Popped: 6
40 >>> 0
41 Popped: 5
42 >>> 0
43 Stack is Empty
44 >>> 0
45 Stack is Empty
46 >>> 4
47 >>> 6
48 >>> 6
49 >>> 7
50 >>> 0
51 Popped: 7
52 >>> 0
53 Popped: 6
54 >>> 0
55 Popped: 6
56 >>> 0
57 Popped: 4
58 >>> 0
59 Stack is Empty
60 >>> 0
61 Stack is Empty
62 >>> -1
63
```

Problem - 3

C Code

```
1
2 /**
3  * Data Structures Assignment - 3
4  * CS 392
5  * Problem 2:
6  *      Write a C program to reverse a string using stack
7  *
8  * Joydeep Mukherjee
9  * CSE, 3rd Sem, 11000116030
10 * GCETTS 2017
11 *
12 **/
13
14 #include <stdio.h>
15 #include <stdlib.h>
16
17
18 typedef struct node_t {
```

```
19  int val;
20  struct node_t *next;
21 }node;
22
23 node *stack;
24
25 void init() {
26     // Create Stack
27     stack = NULL;
28 }
29 void push(int elem) {
30     // Allocate New node
31     node *new = malloc(sizeof(node));
32     // Check for memory fault
33     if(!new) {
34         printf("Memory Fault\n");
35         return;
36     }
37     new->val = elem;
38     new->next = stack;
39
40     stack = new;
41 }
42 int pop() {
43     int temp;
44     // Check for empty stack
45     if(!stack) {
46         //printf("Stack is Empty\n");
47         return 0;
48     }
49     temp = stack->val;
50     stack = stack->next;
51
52     return temp;
53 }
54
55
56 int main(void ){
57     char c;
58
59     init();
60     printf("Enter String:\n>>>");
61     while((c = getchar()) != '\n') {
62         push(c);
63     }
64     printf("Reversed String: ");
65     while(c = pop())
66         printf("%c",c);
67     printf("\n");
68
69
70     return 0;
71 }
72
73
```

Output

```
1 → ./3_3
2 Enter String:
```

```
3 >>>joydeep mukherjee
4 Reversed String: eejrehkum peedyoj
5
```

Problem - 4

C Code

```
1 /**
2  * Data Structures Assignment - 3
3  * CS 392
4  * Problem 2:
5  *     Write a C program to evaluate a postfix expression
6  *
7  * Joydeep Mukherjee
8  * CSE, 3rd Sem, 11000116030
9  * GCETTS 2017
10 *
11 **/
12
13 #include <stdio.h>
14 #include <stdlib.h>
15
16
17 typedef struct node_t {
18     int val;
19     struct node_t *next;
20 }node;
21
22 node *stack;
23
24 void init() {
25     // Create Stack
26     stack = NULL;
27 }
28 void push(int elem) {
29     // Allocate New node
30     node *new = malloc(sizeof(node));
31     // Check for memory fault
32     if(!new) {
33         printf("Memory Fault\n");
34         return;
35     }
36     new->val = elem;
37     new->next = stack;
38
39     stack = new;
40 }
41 int pop() {
42     int temp;
43     // Check for empty stack
44     if(!stack) {
45         //printf("Stack is Empty\n");
46         return 0;
47     }
48     temp = stack->val;
49     stack = stack->next;
50
51     return temp;
```

```

52 }
53
54
55 int main(void ){
56     char c;
57     int a,b;
58     init();
59     printf("Enter Postfix Expression:\n>>>");
60     while((c = getchar()) != '\n') {
61         if(c >= 48 && c < 58) {
62             //printf("Pushed: %d\n",c-48);
63             push(c-48);
64         }
65         else if (c != ' '){
66             b = pop();
67             a = pop();
68             //printf("Popped: %d %d\n",a,b);
69             switch (c) {
70                 case '+':
71                     push(a+b);
72                     break;
73                 case '-':
74                     push(a-b);
75                     break;
76                 case '*':
77                     push(a*b);
78                     break;
79                 case '/':
80                     push(a/b);
81                     break;
82                 default:
83                     break;
84             }
85         }
86     }
87     printf("Result of Expression: %d\n",pop());
88
89     return 0;
90 }

```

Output

```

1 → ./3_4
2 Enter Postfix Expression:
3 >>>1 2 + 6 9 - /
4 Result of Expression: -1

```

Problem - 5

C Code

```

1 /**
2  * Data Structures Assignment - 3
3  * CS 392
4  * Problem 5:
5  *     Write a C program to convert an infix expression to a postfix expression
6  *
7  * Joydeep Mukherjee

```



```
8  * CSE,3rd Sem, 11000116030
9  * GCETTS 2017
10 *
11 **/
12
13 #include <stdio.h>
14
15 char stack[20];
16 int top = -1;
17 void push(char x)
18 {
19     stack[++top] = x;
20 }
21
22 char pop()
23 {
24     if(top == -1)
25         return -1;
26     else
27         return stack[top--];
28 }
29
30 int priority(char x)
31 {
32     if(x == '(')
33         return 0;
34     if(x == '+' || x == '-')
35         return 1;
36     if(x == '*' || x == '/')
37         return 2;
38 }
39
40 int main(void)
41 {
42     char exp[20];
43     char *e, x;
44     printf("Enter the expression :: ");
45     scanf("%s",exp);
46     e = exp;
47     while(*e != '\0')
48     {
49         if(isalnum(*e))
50             printf("%c",*e);
51         else if(*e == '(')
52             push(*e);
53         else if(*e == ')')
54         {
55             while((x = pop()) != '(')
56                 printf("%c", x);
57         }
58         else
59         {
60             while(priority(stack[top]) >= priority(*e))
61                 printf("%c",pop());
62             push(*e);
63         }
64         e++;
65     }
66     while(top != -1)
67     {
68         printf("%c",pop());
```

```
69     }
70 }
71
```

Problem - 6

C Code

```
1  /**
2  * Data Structures Assignment - 3
3  * CS 392
4  * Problem 6:
5  *      Write a C program to implement a circular queue
6  * Joydeep Mukherjee
7  * CSE,3rd Sem, 11000116030
8  * GCETTS 2017
9  *
10 **/
11
12 #include <stdio.h>
13
14 #define SIZE 5
15
16 int items[SIZE];
17 int front = -1, rear = -1;
18
19 int isFull()
20 {
21     if( (front == rear + 1) || (front == 0 && rear == SIZE-1)) return 1;
22     return 0;
23 }
24
25 int isEmpty()
26 {
27     if(front == -1) return 1;
28     return 0;
29 }
30
31 void enqueue(int element)
32 {
33     if(isFull()) printf("\n Queue is full!! \n");
34     else
35     {
36         if(front == -1) front = 0;
37         rear = (rear + 1) % SIZE;
38         items[rear] = element;
39         printf("\n Inserted -> %d", element);
40     }
41 }
42
43
44 int dequeue()
45 {
46     int element;
47     if(isEmpty()) {
48         printf("\n Queue is empty !! \n");
```

```
49     return(-1);
50 } else {
51     element = items[front];
52     if (front == rear){
53         front = -1;
54         rear = -1;
55     } /* Q has only one element, so we reset the queue after dequeing it. ? */
56     else {
57         front = (front + 1) % SIZE;
58     }
59     printf("\n Deleted element -> %d \n", element);
60     return(element);
61 }
62 }
63 }
64
65
66
67
68 void display()
69 {
70     int i;
71     if(isEmpty()) printf(" \n Empty Queue\n");
72     else
73     {
74         printf("\n Front -> %d ",front);
75         printf("\n Items -> ");
76         for( i = front; i!=rear; i=(i+1)%SIZE) {
77             printf("%d ",items[i]);
78         }
79         printf("%d ",items[i]);
80         printf("\n Rear -> %d \n",rear);
81     }
82 }
83
84 int main()
85 {
86     // Fails because front = -1
87     deQueue();
88
89     enqueue(1);
90     enqueue(2);
91     enqueue(3);
92     enqueue(4);
93     enqueue(5);
94
95     // Fails to enqueue because front == 0 && rear == SIZE - 1
96     enqueue(6);
97
98     display();
99     deQueue();
100
101     display();
102
103     enqueue(7);
104     display();
105
106     // Fails to enqueue because front == rear + 1
107     enqueue(8);
108
109     return 0;
```

110 }

Problem - 7

C Code

```
1  /**
2  * Data Structures Assignment - 3
3  * CS 392
4  * Problem 7:
5  *      Write a C program to implement an restricted deque
6  * Joydeep Mukherjee
7  * CSE,3rd Sem, 11000116030
8  * GCETTS 2017
9  *
10 **/
11 #include <stdio.h>
12
13 #define MAX 30
14
15 typedef struct dequeue
16 {
17     int data[MAX];
18     int rear,front;
19 }dequeue;
20
21 void initialize(dequeue *p);
22 int empty(dequeue *p);
23 int full(dequeue *p);
24 void enqueueR(dequeue *p,int x);
25 void enqueueF(dequeue *p,int x);
26 int dequeueF(dequeue *p);
27 int dequeueR(dequeue *p);
28 void print(dequeue *p);
29
30 void main()
31 {
32     int i,x,op,n;
33     dequeue q;
34
35     initialize(&q);
36
37     do
38     {
39
40         printf("\n1.Create\n2.Insert(rear)\n3.Insert(front)\n4.Delete(rear)\n5.Delete(front)");
41         printf("\n6.Print\n7.Exit\n\nEnter your choice:");
42         scanf("%d",&op);
43
44         switch(op)
45         {
46             case 1: printf("\nEnter number of elements:");
47                     scanf("%d",&n);
48                     initialize(&q);
49                     printf("\nEnter the data:");
50
51                     for(i=0;i<n;i++)
52                     {
53                         scanf("%d",&x);
```

```
53         if(full(&q))
54         {
55             printf("\nQueue is full!!");
56             exit(0);
57         }
58         enqueueR(&q,x);
59     }
60     break;
61
62     case 2: printf("\nEnter element to be inserted:");
63             scanf("%d",&x);
64
65             if(full(&q))
66             {
67                 printf("\nQueue is full!!");
68                 exit(0);
69             }
70
71             enqueueR(&q,x);
72             break;
73
74     case 3: printf("\nEnter the element to be inserted:");
75             scanf("%d",&x);
76
77             if(full(&q))
78             {
79                 printf("\nQueue is full!!");
80                 exit(0);
81             }
82
83             enqueueF(&q,x);
84             break;
85
86     case 4: if(empty(&q))
87             {
88                 printf("\nQueue is empty!!");
89                 exit(0);
90             }
91
92             x=dequeueR(&q);
93             printf("\nElement deleted is %d\n",x);
94             break;
95
96     case 5: if(empty(&q))
97             {
98                 printf("\nQueue is empty!!");
99                 exit(0);
100            }
101
102            x=dequeueF(&q);
103            printf("\nElement deleted is %d\n",x);
104            break;
105
106     case 6: print(&q);
107             break;
108
109     default: break;
110 }
111 }while(op!=7);
112 }
113
```

```
114 void initialize(dequeue *P)
115 {
116     P->rear=-1;
117     P->front=-1;
118 }
119
120 int empty(dequeue *P)
121 {
122     if(P->rear==-1)
123         return(1);
124
125     return(0);
126 }
127
128 int full(dequeue *P)
129 {
130     if((P->rear+1)%MAX==P->front)
131         return(1);
132
133     return(0);
134 }
135
136 void enqueueR(dequeue *P,int x)
137 {
138     if(empty(P))
139     {
140         P->rear=0;
141         P->front=0;
142         P->data[0]=x;
143     }
144     else
145     {
146         P->rear=(P->rear+1)%MAX;
147         P->data[P->rear]=x;
148     }
149 }
150
151 void enqueueF(dequeue *P,int x)
152 {
153     if(empty(P))
154     {
155         P->rear=0;
156         P->front=0;
157         P->data[0]=x;
158     }
159     else
160     {
161         P->front=(P->front-1+MAX)%MAX;
162         P->data[P->front]=x;
163     }
164 }
165
166 int dequeueF(dequeue *P)
167 {
168     int x;
169
170     x=P->data[P->front];
171
172     if(P->rear==P->front)    //delete the last element
173         initialize(P);
174     else
```

```
175         P->front=(P->front+1)%MAX;
176
177     return(x);
178 }
179
180 int dequeueR(dequeue *P)
181 {
182     int x;
183
184     x=P->data[P->rear];
185
186     if(P->rear==P->front)
187         initialize(P);
188     else
189         P->rear=(P->rear-1+MAX)%MAX;
190
191     return(x);
192 }
193
194 void print(dequeue *P)
195 {
196     if(empty(P))
197     {
198         printf("\nQueue is empty!!");
199         exit(0);
200     }
201
202     int i;
203     i=P->front;
204
205     while(i!=P->rear)
206     {
207         printf("\n%d",P->data[i]);
208         i=(i+1)%MAX;
209     }
210
211     printf("\n%d\n",P->data[P->rear]);
212 }
```

Output

```
1 1.Create
2 2.Insert(rear)
3 3.Insert(front)
4 4.Delete(rear)
5 5.Delete(front)
6 6.Print
7 7.Exit
8 Enter your choice:1
9
10 Enter number of elements:3
11
12 Enter the data:4 6 7
13
14 1.Create
15 2.Insert(rear)
16 3.Insert(front)
17 4.Delete(rear)
18 5.Delete(front)
19 6.Print
```

```
20 7.Exit
21
22 Enter your choice:6
23
24 4
25 6
26 7
27
28 1.Create
29 2.Insert(rear)
30 3.Insert(front)
31 4.Delete(rear)
32 5.Delete(front)
33 6.Print
34 7.Exit
35
36 Enter your choice:4
37
38 Element deleted is 7
39
40 1.Create
41 2.Insert(rear)
42 3.Insert(front)
43 4.Delete(rear)
44 5.Delete(front)
45 6.Print
46 7.Exit
47
48 Enter your choice:7
```