VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Master of Computer Applications
(SITE)

ITA5002 - Problem Solving with DSA

Digital Assignment
Challenging Exercises (1-7)

Rishav Nath Pati
22MCA0114

1. Students of a Programming class arrive to submit assignments. Their register numbers are stored in a LIFO list in the order in which the assignments are submitted. Write a program using an array to display the register number of the ten students who submitted first. Register number of the ten students who submitted first will be at the bottom of the LIFO list. Hence pop out the required number of elements from the top so as to retrieve and display the first 10 students.

```
#include <stdio.h>
#define max 100

int top = -1;
char a[max][100];

int isempty();
int isfull();
void push(char x[100]);
void pop();
void display();

int main()
{
	char x[100];
	int ch = 1, n;
	while (ch != 4)
	{
	printf("1.PUSH\n");
	printf("2.POP\n");
	printf("3.DISPLAY\n");
	printf("4.EXIT\n");
	printf("Enter your choice: ");
	scanf("%d", &ch);
	switch (ch)
	{
	case 1:
	printf("Enter the registration no. of student: ");
	scanf("%s", x);
	push(x);
	break;
	case 2:
	pop();
	break;
	case 3:
	display();
	break;
	}
	}
	printf("Enter the first n number of required record\n");
	scanf("%d", &n);
	while (top >= n)
	{
	pop();
	}
	printf("The first %d students are\n", n);
	display();
	return 0;
}
int isempty()
{
	if (top == -1)
	return 1;
	else
	return 0;
}
int isfull()
{
	if (top == max - 1)
	return 1;
	else
	return 0;
```

```
}
void push(char x[100])
{
        int i;
        if (isfull())
        printf("stack is full\n");
        else
        {
        top++;
        for (i = 0; i < 100; i++)
        {
        a[top][i] = x[i];
        }
        }
}
void pop()
{
        if (isempty())
        {
        printf("stack is empty\n");
        }
        else
        {
        printf("deleted element is %s\n", a[top]);
        top--;
        }
}
void display()
{
        int i;
        if (isempty())
        printf("stack is empty\n");
        else
        {
        for (i = 0; i <= top; i++)
        printf("%s\n", a[i]);
        }
}
```

OUTPUT:

```
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 12
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 23
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 34
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 45
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 67
```

1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 89
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 56
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 32
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 43
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 65
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 87
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 98
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 3
12
23
34
45
67
89
56
32
43
65
87
98
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 44
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1

```
Enter the registration no. of student: 55
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 1
Enter the registration no. of student: 66
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 3
12
23
34
45
67
89
56
32
43
65
87
98
44
55
66
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter your choice: 4
Enter the first n number of required record
10
deleted element is 66
deleted element is 55
deleted element is 44
deleted element is 98
deleted element is 87
The first 10 students are
12
23
34
45
67
89
56
32
43
65
```

2. To facilitate thorough net surfing, any web browser has back and forward buttons that allow the user to move backward and forward through a series of web pages. To allow the user to move both forward and backward two stacks are employed. When the user presses the back button, the link to the current web page is stored on a separate stack for the forward button. As the user moves backward through a series of previous pages, the link to each page is moved in turn from the back to the forward stack. When the user presses the forward button, the action is the reverse of the back button. Now the item from the forward stack is popped, and becomes the current web page. The previous web page is pushed on the back stack. Simulate the functioning of these buttons using array implementation of Stack. Also provide options for displaying the contents of both the stacks whenever required.

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

#define n 20
char stack[n][100];
char backs[n][100];
int top1 = -1;
int top2 = -1;
void push1(char str[])
{
   top1++;
   strcpy(stack[top1], str);
}
void push2(char str[])
{
   top2++;
   strcpy(backs[top2], str);
}
void pop1()
{
   top2++;
   strcpy(backs[top2], stack[top1]);
   top1--;
}
int pop2()
{
   top1++;
   strcpy(stack[top1], backs[top2]);
   top2--;
   printf("Done");
   return 0;
}
```

```
void display()
{
    if (top1 == -1)
        printf("Nothing");
    for (int i = 0; i <= top1; i++)
    {
        printf("%s", stack[i]);
    }
}
int main()
{
    int ch;
    char str[100];
    do
    {
        printf("\n\
        1. Push\n\
        2. Backward \n\
        3. Forward\n\
        4. Display\n\
        5. Exit \n");
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch (ch)
        {

        case 1:
            printf("Enter URL to visit: ");
            scanf("%s", str);
            push1(str);
            break;
        case 2:
            pop1();
            break;
        case 3:
            pop2();
            break;
        case 4:
            display();
            break;
        case 5:
            printf("Exit");
            break;
        default:
```

```
            printf("Wrong choice");
            break;
        }
    } while (ch != 5);
    return 0;
}
```

OUTPUT:

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 1

Enter URL to visit: page1

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 1

Enter URL to visit: page2

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 1

Enter URL to visit: page3

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 4

page1page2page3

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 1

Enter URL to visit: page4

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 3

Done

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 4

page1page2page3page4

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 2

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 4

page1page2page3page4

1. Push
2. Backward
3. Forward

4. Display
5. Exit

Enter your choice: 2

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 2

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 4

page1page2

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 3

Done

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 4

page1page2page3

1. Push
2. Backward
3. Forward
4. Display
5. Exit

Enter your choice: 5

Exit%

3. Write a program to implement a 3-stacks of size ‘m’ in an array of size ‘n’ with all the basic operations such as IsEmpty(i), Push(i), Pop(i), IsFull(i) where ‘i’ denotes the stack number (1,2,3), m n/3. Stacks are not overlapping each other. Leftmost stack is facing the left direction and the other two stacks are facing in the right direction.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

int stack[MAX];
int top1 = -1;
int top2 = MAX / 3;
int top3 = (MAX / 3) * 2;

void push(int stackNum, int data)
{
    if (stackNum == 1)
    {
        if (top1 == MAX / 3 - 1)
        {
            printf("Stack is full");
            return;
        }
        stack[++top1] = data;
    }
    else if (stackNum == 2)
    {
        if (top2 == (MAX / 3) * 2 - 1)
        {
            printf("Stack is full");
            return;
        }
        stack[++top2] = data;
    }
    else if (stackNum == 3)
    {
        if (top3 == MAX - 1)
        {
            printf("Stack is full");
            return;
        }
```

```
        stack[++top3] = data;
    }
    else
    {
        printf("Invalid stack number");
    }
}

int pop(int stackNum)
{
    if (stackNum == 1)
    {
        if (top1 == -1)
        {
            printf("Stack is empty");
            return -1;
        }
        return stack[top1--];
    }
    else if (stackNum == 2)
    {
        if (top2 == MAX / 3)
        {
            printf("Stack is empty");
            return -1;
        }
        return stack[top2--];
    }
    else if (stackNum == 3)
    {
        if (top3 == (MAX / 3) * 2)
        {
            printf("Stack is empty");
            return -1;
        }
        return stack[top3--];
    }
    else
    {
        printf("Invalid stack number");
        return -1;
    }
}
```

```
int isEmpty(int stackNum)
{
    if (stackNum == 1)
    {
        if (top1 == -1)
        {
            return 1;
        }
        return 0;
    }
    else if (stackNum == 2)
    {
        if (top2 == MAX / 3)
        {
            return 1;
        }
        return 0;
    }
    else if (stackNum == 3)
    {
        if (top3 == (MAX / 3) * 2)
        {
            return 1;
        }
        return 0;
    }
    else
    {
        printf("Invalid stack number");
        return -1;
    }
}

int isFull(int stackNum)
{
    if (stackNum == 1)
    {
        if (top1 == MAX / 3 - 1)
        {
            return 1;
        }
        return 0;
    }
    else if (stackNum == 2)
```

```
    {
        if (top2 == (MAX / 3) * 2 - 1)
        {
            return 1;
        }
        return 0;
    }
    else if (stackNum == 3)
    {
        if (top3 == MAX - 1)
        {
            return 1;
        }
        return 0;
    }
    else
    {
        printf("Invalid stack number");
        return -1;
    }
}

int main()
{

    while (1)
    {
        int choice;
        printf("\n1. Push \n2. Pop \n3. IsEmpty \n4. IsFull \n5. Exit \nEnter your choice: ");
        scanf("%d", &choice);

        if (choice == 1)
        {
            int stackNum, data;
            printf("Enter stack number: ");
            scanf("%d", &stackNum);
            printf("Enter data: ");
            scanf("%d", &data);
            push(stackNum, data);
        }
        else if (choice == 2)
        {
            int stackNum;
            printf("Enter stack number: ");
```

```
            scanf("%d", &stackNum);
            printf("Popped element: %d", pop(stackNum));
        }
        else if (choice == 3)
        {
            int stackNum;
            printf("Enter stack number: ");
            scanf("%d", &stackNum);
            printf("Is empty: %d", isEmpty(stackNum));
        }
        else if (choice == 4)
        {
            int stackNum;
            printf("Enter stack number: ");
            scanf("%d", &stackNum);
            printf("Is full: %d", isFull(stackNum));
        }
        else if (choice == 5)
        {
            break;
        }
        else
        {
            printf("Invalid choice");
        }
    }
}
```

OUTPUT:

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Exit

Enter your choice: 1

Enter stack number: 1

Enter data: 12

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Exit

Enter your choice: 1

Enter stack number: 1

Enter data: 13

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Exit

Enter your choice: 1

Enter stack number: 2

Enter data: 21

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Exit

Enter your choice: 2

Enter stack number: 2

Popped element: 21

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Exit

Enter your choice: 22

Invalid choice

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Exit

Enter your choice: 1

Enter stack number: 3

Enter data: 33

1. Push
2. Pop
3. IsEmpty
4. IsFull
5. Exit

Enter your choice: 1

Enter stack number: 3

Enter data: 34

1. Push

2. Pop

3. IsEmpty

4. IsFull

5. Exit

Enter your choice: 2

Enter stack number: 3

Popped element: 34

1. Push

2. Pop

3. IsEmpty

4. IsFull

5. Exit

Enter your choice: 3

Enter stack number: 3

Is empty: 0

1. Push

2. Pop

3. IsEmpty

4. IsFull

5. Exit

Enter your choice: 5

4. Most of the bugs in scientific and engineering applications are due to improper usage of precedence order in arithmetic expressions. Thus it is necessary to use an appropriate notation that would evaluate the expression without taking into account the precedence order and parenthesis.

Write a program to convert the given arithmetic expression into

- Reverse Polish notational
- Polish notation

```
#include <stdio.h>
#include <math.h>
float scannum(char ch)
{
   int value;
   value = ch;
   return (float)(value - '0');
}
int isOperator(char ch)
{
   if (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^')
      return 1;
   return -1;
}
```

```
int isOperand(char ch)
{
    if (ch >= '0' && ch <= '9')
        return 1;
    return -1;
}
float operation(int a, int b, char op)
{
    if (op == '+')
        return b + a;
    else if (op == '-')
        return b - a;
    else if (op == '*')
        return b * a;
    else if (op == '/')
        return b / a;
    else if (op == '^')
        return pow(b, a);
    else
        return 0;
}
float postfix(char *postfix)
{
    int a, b;
    int stk[100];
    int top = -1;
    char *it;
    for (it = postfix; *it != '\0'; it++)
    {
        if (isOperator(*it) != -1)
        {
            a = stk[top];
            top--;
            b = stk[top];
            top--;
            top++;
            stk[top] = operation(a, b, *it);
        }
        else if (isOperand(*it) > 0)
        {
            top++;
            stk[top] = scannum(*it);
        }
    }
```

```
    return stk[top];
}
int main()
{

    char post[] = "45+78/35*2";
    printf("%s = ", post);
    printf("%f", postfix(post));
}
```

45+78/35*2 = 2.000000

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

double evaluateprefix(char *prefixExp)
{

    char operendStack[100];
    int top = -1;

    for (int i = strlen(prefixExp) - 1; i >= 0; i--)
    {

        if (isdigit(prefixExp[i]))
            operendStack[++top] = prefixExp[i] - '0';
        else
        {
            double o1 = operendStack[top--];
            double o2 = operendStack[top--];
            if (prefixExp[i] == '+')
                operendStack[++top] = o1 + o2;
            else if (prefixExp[i] == '-')
                operendStack[++top] = o1 - o2;
            else if (prefixExp[i] == '*')
                operendStack[++top] = o1 * o2;
            else if (prefixExp[i] == '/')
                operendStack[++top] = o1 / o2;
            else
            {
```

```
            printf("Invalid Expression");
            return -1;
          }
        }
    }
    return operendStack[top];
}

int main()

{
    char prefix[] = "*+54-23";
    printf("%s= %f", prefix, evaluateprefix(prefix));
}
```

```
*+54-23= -9.000000
```

5. In a theme park, the Roller-Coaster ride is started only when a good number of riders line up in the counter (say 20 members). When the ride proceeds with these 20 members, a new set of riders will line up in the counter. This keeps continuing. Implement the above scenario of lining up and processing using arrays with Queue ADT.

```
#include <stdio.h>
#define n 20
int q[n];
int front = -1, rear = -1;

void enqueue(int data)
{
    if (rear == 20)
    {
        printf("waiting for ride");
    }
    rear++;
    q[rear] = data;
}
```

```c
void dequeue()
{
    if (rear == -1 && front == -1)
        printf("ride empty");
    front++;
}

void display()
{
    if (rear == -1)
        printf("No one up for a ride");
    int i = front;
    while (i <= rear)
    {
        printf("%d", q[i]);

        i++;
    }
}

int main()
{
    int ch;
    int v;
    do
    {
        printf("\n\
        1. Insert\n\
        2. Delete\n\
        3. Dispaly\n\
        4. to Exit\n");
        printf("enter your choice: ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            for (int i = 0; i < n; i++)
            {
                printf("Enter rider: ");
                scanf("%d", &v);
                enqueue(v);
            }
            printf("wait till the next ride starts...");
            break;
```

```
        case 2:
            for (int i = 19; i > 0; i++)
            {
                dequeue();
            }
            printf("Entering the next ride");
            break;
        case 3:
            display();
            break;

        case 4:
            printf("Exiting.....");
            break;
        default:
            printf("enter the correct choice");
        }
    } while (ch != 4);
}
```

OUTPUT:

```
        1. Insert
        2. Delete
        3. Dispaly
        4. to Exit
enter your choice: 1
Enter rider: 1
Enter rider: 2
Enter rider: 3
Enter rider: 4
Enter rider: 5
Enter rider: 6
Enter rider: 7
Enter rider: 8
Enter rider: 9
Enter rider: 10
wait till the next ride starts...
        1. Insert
        2. Delete
        3. Dispaly
        4. to Exit
enter your choice: 3
912345678910
```

1. Insert
2. Delete
3. Dispaly
4. to Exit

enter your choice: 1

Enter rider: 1

Enter rider: 2

Enter rider: 3

Enter rider: 4

Enter rider: 5

Enter rider: 6

Enter rider: 7

Enter rider: 8

Enter rider: 9

Enter rider: 10

wait till the next ride starts...

1. Insert
2. Delete
3. Dispaly
4. to Exit

enter your choice: 2

1

2Entering the next ride

1. Insert
2. Delete
3. Dispaly
4. to Exit

enter your choice: Enter rider:

Enter rider: 3

waiting for rideEnter rider: 3

Enter rider: 4

Enter rider: 5

Enter rider: 6

Enter rider: 7

Enter rider: 8

Enter rider: 9

Enter rider: 1

wait till the next ride starts...

1. Insert
2. Delete
3. Dispaly
4. to Exit

enter your choice: 4

6. When burning a DVD it is essential that the laser beam burning pits onto the surface is constantly fed with data, otherwise the DVD fails. Most leading DVD burn applications make use of a circular buffer to stream data from the hard disk onto the DVD. The first part, the 'writing process' fills up a circular buffer with data, then the 'burning process' begins to read from the buffer as the laser beam burns pits onto the surface of the DVD. If the buffer starts to become empty, the application should continue filling up the emptied space in the buffer with new data from the disk. Implement this scenario using Circular Queue.

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

#define MAX 40

int rear = -1;
int front = -1;
char q[MAX];
char b[30];
char at[] = {"abcdefghij"};
int i = 0, j = 0;

int isempty()
{
   if (front == -1)
      return 1;
   else
      return 0;
}
int isfull()
{
   if ((rear + 1) % 10 == front)
      return 1;
   else
      return 0;
}
void write()
{
```

```
    if (isfull())
        printf("Full");
    else
    {
        if (front == -1)
            front = 0;
        rear = (rear + 1) % 10;
        q[rear] = at[i++];
    }
}
void disp()
{
    int j;
    printf("Output");
    for (j = 0; j < 20; j++)
        printf("%c", b[j]);
    printf("\n");
}
void read()
{
    if (rear == front)
        printf("Disk Empty");
    else
    {
        char d[54];
        d[j] = q[front];
        disp();
        front = (front + 1) % 10;

        j++;
    }
}
void display()
{
    int i;
    printf("In Buffer\n");
    for (i = 0; i < MAX; i++)
        printf("%c", q[i]);
    printf("\n");
}
int main()
{
    while (isfull() != 1)
    {
```

```
        write();
    }
    int ch;
    display();
    while (ch != 2)
    {
        printf("1 to read\n");
        printf("2 to exit\n");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            read();

            write();
            display();
        }
    }
}
```

OUTPUT:
In Buffer
bcdefghij
1 to read
2 to exit
1
Output
In Buffer
cdefghij
1 to read
2 to exit
1
Output
In Buffer
defghij
1 to read
2 to exit
1
Output
In Buffer
efghij
1 to read
2 to exit
1
Output
In Buffer
fghij
1 to read
2 to exit

1
Output
In Buffer
ghij
1 to read
2 to exit
1
Output
In Buffer
hij
1 to read
2 to exit
1
Output
In Buffer
ij
1 to read
2 to exit
1
Output
In Buffer
j
1 to read
2 to exit
1
Output
In Buffer

1 to read
2 to exit
1
Output
In Buffer

1 to read
2 to exit

7. Assume FLAMES game that tests for relationship has to be implemented using a dynamic structure. The letters in the FLAMES stand for Friends, Love, Affection, Marriage, Enmity and Sister. Initially store the individual letters of the word 'flames' in the nodes of the dynamic structure. Given the count of the number of uncommon letters in the two names 'n', write a program to delete every nth node in it, till it is left with a single node. If the end of the dynamic structure is reached while counting, resume the counting from the beginning. Display the letter that still remains and the corresponding relationship.

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

struct node
```

```
{
    char data;
    struct node *next;
};

struct node *head = NULL;

void insert(char data)
{
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = data;
    newNode->next = NULL;
    if (head == NULL)
    {
        head = newNode;
        head->next = head;
    }
    else
    {
        struct node *temp = head;
        while (temp->next != head)
        {
            temp = temp->next;
        }
        temp->next = newNode;
        newNode->next = head;
    }
}

void display()
{
    struct node *temp = head;
    while (temp->next != head)
    {
        printf("%c ", temp->data);
        temp = temp->next;
    }
    printf("%c ", temp->data);
}

void deleteNode(int n)
{
    struct node *temp = head;
    struct node *prev = NULL;
```

```
    int count = 1;
    while (temp->next != temp)
    {
        if (count == n)
        {
            prev->next = temp->next;
            free(temp);
            temp = prev->next;
            count = 1;
        }
        else
        {
            prev = temp;
            temp = temp->next;
            count++;
        }
    }
    head = temp;
}

int main()
{
    char flames[] = "flames";
    for (int i = 0; i < strlen(flames); i++)
    {
        insert(flames[i]);
    }

    printf("The letters in the word flames are: ");
    display();

    int n;

    printf("Enter the first name: ");
    char name1[100];
    scanf("%s", name1);

    printf("Enter the second name: ");
    char name2[100];
    scanf("%s", name2);

    int count = 0;
    for (int i = 0; i < strlen(name1); i++)
    {
```

```
        for (int j = 0; j < strlen(name2); j++)
        {
            if (name1[i] == name2[j])
            {
                name1[i] = ' ';
                name2[j] = ' ';
                count++;
                break;
            }
        }
    }

    n = strlen(name1) + strlen(name2) - 2 * count;
    deleteNode(n);
    printf("The relationship is: ");
    display();
    return 0;
}
```

OUTPUT:
The letters in the word flames are: f l a m e s
Enter the first name: harini

Enter the second name: harish
The relationship is: e