QUESTION 1: Write a program to implement two stacks in one array

ANSWER:

```
//To implement two stacks using single array

#include <stdio.h>
#include <stdlib.h>
#define MAX 6
void insert(int *, int *top1, int *top2, int data, int stackno);
int pop(int *, int *top1, int *top2, int stackno);
int top(int *, int *top1, int *top2, int stackno);
void display(int *, int top1, int top2, int stackno);
int main()
{
    system("cls");
    int a[MAX];
    static int top1, top2; //top1 -> for stack 1 and top2 for stack 2;
    top1 = -1;
    top2 = MAX;
    printf("\n\n\nEnter 1 for insetion\nEnter 2 for deletion \nEnter 3 for display \nEnt
er 4 for view top of stack\nEnter any other for exit");
    while (1)
    {
        int choice;
        printf("\nEnter your choice:");
        scanf("%d", &choice);
        if (choice == 1)
        {
            int data, stackno;
            printf("Enter the data:");
            scanf("%d", &data);
            printf("Enter the stack number:");
            scanf("%d", &stackno);
            insert(a, &top1, &top2, data, stackno);
        }
        else if (choice == 2)
        {
            int rem, stackno;
            printf("Enter the stack number:");
            scanf("%d", &stackno);
            rem = pop(a, &top1, &top2, stackno);
            printf("\nThe removed element is %d", rem);
        }
        else if (choice == 3)
```

```
        {
            int stackno;
            printf("Enter the stack number:");
            scanf("%d", &stackno);
            display(a, top1, top2, stackno);
        }
        else if (choice == 4)
        {
            int topelement, stackno;
            printf("Enter the stack number:");
            scanf("%d", &stackno);
            topelement = top(a, &top1, &top2, stackno);
            printf("\nThe topmost element is %d", topelement);
        }
        else
        {
            printf("\nEnd of the program");
            break;
        }
    }
    return 0;
}

void insert(int *a, int *top1, int *top2, int data, int stackno)
{
    if (stackno == 1)
    {
        (*top1)++;
        //checking for full stack
        if (*top1 == *top2)
        {
            printf("\nStack is Full");
            (*top1)--;
            return;
        }
        else
        {
            a[*top1] = data;
            printf("\nData had been inserted");
            return;
        }
    }
    else if (stackno == 2)
    {
        (*top2)--;
        //checking for full stack
        if (*top2 == *top1)
        {
```

```
            printf("\nStack is Full");
            (*top2)++;
            return;
        }
        else
        {
            a[*top2] = data;
            printf("\nData had been inserted");
            return;
        }
    }
    else
    {
        printf("\nWrong stack number entered");
    }
}
void display(int *a, int top1, int top2, int stackno)
{
    if (stackno == 1)
    {
        //checking for empty stack
        if (top1 == -1)
        {
            printf("\nThe stack is empty");
            return;
        }

        int i = top1;
        printf("\nThe stack is\n");
        for (; i >= 0; i--)
        {
            printf("%d\t", a[i]);
        }
        return;
    }
    else if (stackno == 2)
    {
        //checking for empty stack
        if (top2 == MAX)
        {
            printf("\nThe stack is empty");
            return;
        }
        int i = top2;
        printf("\nThe stack is\n");
        for (; i <= MAX - 1; i++)
        {
            printf("%d\t", a[i]);
```

```
        }
        return;
    }
    else
    {

        printf("\nWrong stack number");
    }
}

int pop(int *a, int *top1, int *top2, int stackno)
{
    if (stackno == 1)
    {
        //checking for empty stack
        if ((*top1) == -1)
        {
            printf("\nThe stack is empty");
            return;
        }
        int removed = a[*top1];
        (*top1)--;
        printf("\nElement had been removed");
        return removed;
    }
    else if (stackno == 2)
    {
        //checking from empty stack
        if ((*top2) == MAX)
        {
            printf("\nThe stack is empty");
            return;
        }
        int removed = a[*top2];
        (*top2)++;
        printf("\nElement had been removed");
        return removed;
    }
    else
    {
        printf("\nWrong stack number");
        return;
    }
}
int top(int *a, int *top1, int *top2, int stackno)
{
    if (stackno == 1)
    {
```

```
        //checking for empty stack
        if ((*top1) == -1)
        {
            printf("\nThe stack is empty");
            return;
        }
        int removed = a[*top1];
        return removed;
    }
    else if (stackno == 2)
    {
        //checking from empty stack
        if ((*top2) == MAX)
        {
            printf("\nThe stack is empty");
            return;
        }
        int removed = a[*top2];
        return removed;
    }
    else
    {
        printf("\nWrong stack number");
        return;
    }
}
```

OUTPUT

```
Enter 1 for insetion
Enter 2 for deletion
Enter 3 for display
Enter 4 for view top of stack
Enter any other for exit
Enter your choice:1
Enter the data:10
Enter the stack number:1

Data had been inserted
Enter your choice:1
Enter the data:20
Enter the stack number:1

Data had been inserted
Enter your choice:1
Enter the data:30
Enter the stack number:2

Data had been inserted
Enter your choice:1
Enter the data:40
Enter the stack number:1

Data had been inserted
Enter your choice:1
Enter the data:50
Enter the stack number:1

Data had been inserted
Enter your choice:1
Enter the data:60
Enter the stack number:2

Data had been inserted
Enter your choice:3
Enter the stack number:1

The stack is
50      40      20      10
```

```
Enter your choice:3
Enter the stack number:2

The stack is
60      30
Enter your choice:2
Enter the stack number:1

Element had been removed
The removed element is 50
Enter your choice:2
Enter the stack number:1

Element had been removed
The removed element is 40
Enter your choice:4
Enter the stack number:1

The topmost element is 20
Enter your choice:4
Enter the stack number:1

The topmost element is 20
Enter your choice:10

End of the program
PS C:\Users\amar\Desktop\data>
```

QUESTION 2) WRITE A PROGRAM TO IMPLEMENT STACK USING LINKED LIST.

ANSWER)

```c
#include <stdio.h>
#include <stdlib.h>
struct node
{
    float data;
    struct node *next;
};

struct stack
{
    struct node *top;
};

void push(struct stack *, float);
float pop(struct stack *);
float top(struct stack *);
void display(struct stack *);
int main()
{
    struct stack s1;
    s1.top = NULL;
```

```
    printf("\n\n\nEnter 1 for insetion\nEnter 2 for deletion \nEnter 3 for display \nEnt
er 4 for view top of stack\nEnter any other for exit");
    while (1)
    {
        int choice;
        printf("\nEnter your choice:");
        scanf("%d", &choice);
        if (choice == 1)
        {
            float data;
            printf("Enter the data:");
            scanf("%f", &data);
            push(&s1,data);
        }
        else if (choice == 2)
        {
            float rem;
            rem = pop(&s1);
            printf("\nThe removed element is %.3f", rem);
        }
        else if (choice == 3)
        {
            display(&s1);
        }
        else if (choice == 4)
        {
            float topelement;
            topelement = top(&s1);
            printf("\nThe topmost element is %.3f", topelement);
        }
        else
        {
            printf("\nEnd of the program");
            break;
        }
    }
    return 0;
}

void push(struct stack *s1, float data)
{
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    if (newnode == NULL)
    {
        printf("\nMemory can't be allocated\n");
        return;
    }
```

```
    newnode->data = data;
    newnode->next = NULL;
    if (s1->top == NULL)
    {
        s1->top = newnode;
    }
    else
    {
        newnode->next = s1->top;
        s1->top = newnode;
    }
}

float pop(struct stack *s1)
{
    float n1;
    if (s1->top == NULL)
    {
        printf("\nEmpty stack");
        exit(1);
    }
    n1 = s1->top->data;
    struct node *n2;
    n2 = s1->top;
    s1->top = n2->next;
    free(n2);
    return n1;
}
float top(struct stack *s1)
{
    float n1;
    if (s1->top == NULL)
    {
        printf("\nEmpty stack");
        exit(1);
    }
    n1 = s1->top->data;
    return n1;
}

void display(struct stack *s1)
{
    if (s1->top == NULL)
    {
        printf("\nEmpty stack\n");
        return;
    }
    struct node *current;
```

```
    current = s1->top;
    while (current != NULL)
    {

        printf("%.3f\t", current->data);

        current = current->next;
    }
}
```

OUTPUT

```
Enter 1 for insetion
Enter 2 for deletion
Enter 3 for display
Enter 4 for view top of stack
Enter any other for exit
Enter your choice:1
Enter 1 for insetion
Enter 2 for deletion
Enter 3 for display
Enter 4 for view top of stack
Enter any other for exit
Enter your choice:1
Enter the data:20

Enter your choice:3
20.000
Enter your choice:1
Enter the data:30

Enter your choice:2

The removed element is 30.000
Enter your choice:3
20.000
Enter your choice:1
Enter the data:50

Enter your choice:4

The topmost element is 50.000
Enter your choice:1
Enter the data:60

Enter your choice:3
60.000  50.000  20.000
Enter your choice:15

End of the program
PS C:\Users\amar\Desktop\data>
```

QUESTION 3) WAP TO CONVERT AN INFIX STRING TO POST FIX AND EVALUATE IT

ANSWER :

```
#define SIZE 50 /* Size of Stack */
#include <ctype.h>
#include <stdio.h>

char s[SIZE];
int top = -1; /* Global declarations */

/* Function to remove spaces from given string */
void RemoveSpaces(char *source)
{
    char *i = source;
    char *j = source;
    while (*j != 0)
    {
        *i = *j++;
        if (*i != ' ')
            i++;
    }
    *i = 0;
}

/* Function for PUSH operation */
void push(char elem)
{
    s[++top] = elem;
}

/* Function for POP operation */
char pop()
{
    return (s[top--]);
}

/* Function for precedence */
int pr(char elem)
{
    switch (elem)
    {
    case '#':
        return 0;
    case '(':
        return 1;
    case '+':
    case '-':
        return 2;
```

```
    case '*':
    case '/':
        return 3;
    }
}

/*
* Function to convert from infix to postfix expression
*/
void infix_to_postfix(char *infix, char *postfix)
{
    char ch, elem;
    int i = 0, k = 0;

    RemoveSpaces(infix);
    push('#');

    while ((ch = infix[i++]) != '\n')
    {
        if (ch == '(')
            push(ch);
        else if (isalnum(ch))
            postfix[k++] = ch;
        else if (ch == ')')
        {
            while (s[top] != '(')
                postfix[k++] = pop();
            elem = pop(); /* Remove ( */
        }
        else
        { /* Operator */
            while (pr(s[top]) >= pr(ch))
                postfix[k++] = pop();
            push(ch);
        }
    }

    while (s[top] != '#') /* Pop from stack till empty */
        postfix[k++] = pop();

    postfix[k] = 0; /* Make postfix as valid string */
}

/*
* Function to evaluate a postfix expression
*/
int eval_postfix(char *postfix)
{
```

```
    char ch;
    int i = 0, op1, op2;
    while ((ch = postfix[i++]) != 0)
    {
        if (isdigit(ch))
            push(ch - '0'); /* Push the operand */
        else
        { /* Operator,pop two operands */
            op2 = pop();
            op1 = pop();
            switch (ch)
            {
            case '+':
                push(op1 + op2);
                break;
            case '-':
                push(op1 - op2);
                break;
            case '*':
                push(op1 * op2);
                break;
            case '/':
                push(op1 / op2);
                break;
            }
        }
    }
    return s[top];
}

void main()
{ /* Main Program */

    char infx[50], pofx[50];
    printf("\nInput the infix expression: ");
    fgets(infx, 50, stdin);

    infix_to_postfix(infx, pofx);

    printf("\nGiven Infix Expression: %sPostfix Expression: %s", infx, pofx);
    top = -1;
    printf("\nResult of evaluation of postfix expression : %d", eval_postfix(pofx));
}
```

```
Input the infix expression: 2*(4+1)

Given Infix Expression: 2*(4+1)
Postfix Expression: 241+*
Result of evaluation of postfix expression : 10
PS C:\Users\amar\Desktop\data>
```

```
Input the infix expression: 2*(3+4*2)

Given Infix Expression: 2*(3+4*2)
Postfix Expression: 2342*+*
Result of evaluation of postfix expression : 22
PS C:\Users\amar\Desktop\data>
```

QUESTION 4a) WRITE A PROGRAM TO IMPLEMENT QUEUE USING ARRAY

ANSWER.

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 5

void enqueue(int *a, int *front, int *rear, int data);
int dequeue(int *a, int *front, int *rear);
void display(int *a, int *front, int *rear);

int main()
{
    int a[MAX];
    int front =-1;
    int rear =-1;
    printf("\n\n\nEnter 1 for insetion\nEnter 2 for deletion \nEnter 3 for display \nEnt
er any other for exit");
    while (1)
    {
        int choice;
        printf("\nEnter your choice:");
        scanf("%d", &choice);
        if (choice == 1)
        {
            int data;
            printf("Enter the data:");
            scanf("%d", &data);
            enqueue(a, &front, &rear, data);
        }
        else if (choice == 2)
        {
            int rem;
            rem = dequeue(a,&front,&rear);
            printf("\nThe removed element is %d", rem);
        }
        else if (choice == 3)
        {
```

```
            display(a, &front, &rear);
        }
        else
        {
            printf("\nEnd of the program");
            break;
        }
    }

    return 0;
}

void enqueue(int *a, int *front, int *rear, int data)
{
    if ((*rear) == -1&&(*front)==-1)
    {
        (*front)++;
        (*rear)++;
    }
    else
        (*rear)++;
    if ((*rear) == MAX)
    {
        printf("\nThe queue is full");
        return;
    }
    a[*rear] = data;
    return;
}
int dequeue(int *a, int *front, int *rear)
{   int rem;
    (*front)++;
    if((*front)>(*rear))
    {
        printf("\nThe queue is empty");
    }
    rem=a[--(*front)];
    (*front)++;
    if(*(front)>*(rear))
    {
        (*front)=-1;
        (*rear)=-1;
    }
    return rem;

}
void display(int *a,int *front,int *rear)
{
```

```
    int i;
    i=(*front);
    if((*front)==-1&&(*rear)==-1)
    {
        printf("\nEmpty queue");
        return;
    }
    printf("\nThe queue is \n");
    for(i=(*front);i<=(*rear);i++)
    {
        printf("%d\t",a[i]);
    }
}
```

OUTPUT

```
Enter 1 for insetion
Enter 2 for deletion
Enter 3 for display
Enter any other for exit
Enter your choice:1
Enter the data:20

Enter your choice:1
Enter the data:30

Enter your choice:1
Enter the data:40

Enter your choice:3

The queue is
20      30      40
Enter your choice:2

Enter the data:20

Enter your choice:1
Enter the data:30

Enter your choice:1
Enter the data:40

Enter your choice:3

The queue is
20      30      40
Enter your choice:2

The removed element is 20
Enter your choice:3

The queue is
30      40
Enter your choice:10

End of the program
PS C:\Users\amar\Desktop\data>
```

QUESTION 4B) WAP TO IMPLEMENT QUEUE USING LINKED LIST.

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

struct node{
```

```
    int data;
    struct node *next;
};

struct queue{
    struct node *front;
};
void enqueue(struct queue *s1,int data);
int dequeue(struct queue *s1);
void display(struct queue *s1);

int main()
{
    struct queue q1;
    q1.front=NULL;
    printf("\n\n\nEnter 1 for insetion\nEnter 2 for deletion \nEnter 3 for display \nEnt
er any other for exit");
    while (1)
    {
        int choice;
        printf("\nEnter your choice:");
        scanf("%d", &choice);
        if (choice == 1)
        {
            int data;
            printf("Enter the data:");
            scanf("%d", &data);
            enqueue(&q1, data);
        }
        else if (choice == 2)
        {
            int rem;
            rem = dequeue(&q1);
            printf("\nThe removed element is %d", rem);
        }
        else if (choice == 3)
        {
            display(&q1);
        }
        else
        {
            printf("\nEnd of the program");
            break;
        }
    }

    return 0;
}
```

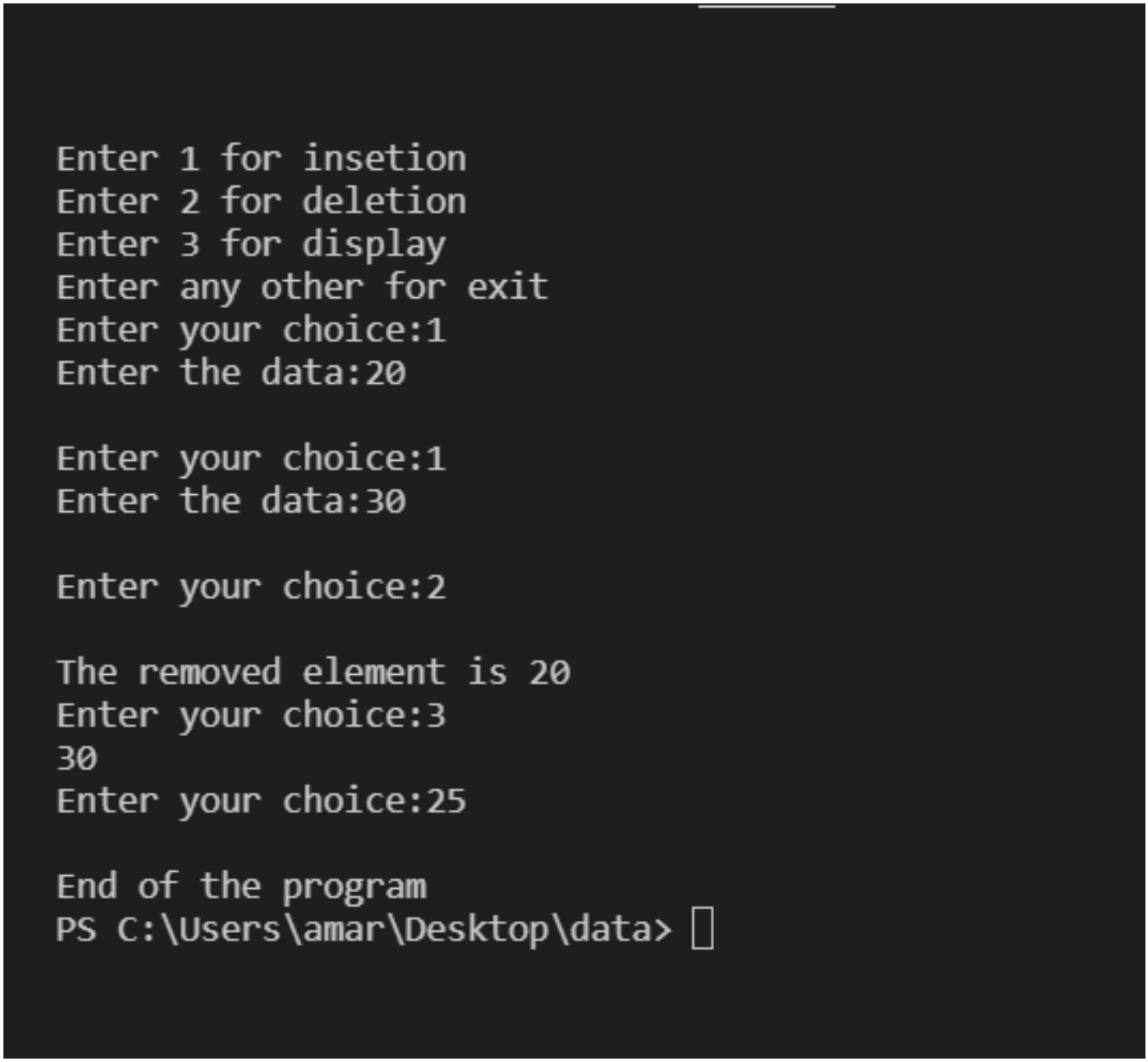
```
void enqueue(struct queue *s1,int data)
{
    struct node *newnode;
    newnode =(struct node *)malloc(sizeof(struct node));
    newnode->data=data;
    newnode->next=NULL;
    if(s1->front==NULL)
    {
        s1->front=newnode;
    }
    else
    {
        struct node *current;
        current = s1->front;
        while(current->next!=NULL)
        {
            current=current->next;
        }
        current->next=newnode;
    }

}
int dequeue(struct queue *s1)
{
    if(s1->front==NULL)
    {
        printf("\nEmpty queue");
        exit(1);
    }
    else
    {
        int rem;
        struct node *removed;
        removed=s1->front;
        s1->front=removed->next;
        rem=removed->data;
        free(removed);
        return rem;
    }

}
void display(struct queue *s1)
{
    struct node *current;
    current=s1->front;
    while(current!=NULL)
    {
        printf("%d\t",current->data);
```

```
        current=current->next;
    }
}
```

OUTPUT



QUESTION 5) TO DISPLAY LINK LIST IN REVERSE ORDER

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

        struct node
{
    int num;
    struct node *next;
};

void create(struct node **);
void reversedisplay(struct node *);
void release(struct node **);
void display(struct node *);

int main()
{
    struct node *p = NULL;
    struct node_occur *head = NULL;
    int n;

    printf("Enter data into the list\n");
    create(&p);
    printf("Displaying the nodes in the list:\n");
    display(p);
    printf("Displaying the list in reverse:\n");
    reversedisplay(p);
    release(&p);

    return 0;
}

void reversedisplay(struct node *head)
{
    if (head != NULL)
    {
        reversedisplay(head->next);
        printf("%d\t", head->num);
    }
}

void create(struct node **head)
{
    int c, ch;
    struct node *temp, *rear;
```

```
    do
    {
        printf("Enter number: ");
        scanf("%d", &c);
        temp = (struct node *)malloc(sizeof(struct node));
        temp->num = c;
        temp->next = NULL;
        if (*head == NULL)
        {
            *head = temp;
        }
        else
        {
            rear->next = temp;
        }
        rear = temp;
        printf("Do you wish to continue [1/0]: ");
        scanf("%d", &ch);
    } while (ch != 0);
    printf("\n");
}

void display(struct node *p)
{
    while (p != NULL)
    {
        printf("%d\t", p->num);
        p = p->next;
    }
    printf("\n");
}

void release(struct node **head)
{
    struct node *temp = *head;
    *head = (*head)->next;
    while ((*head) != NULL)
    {
        free(temp);
        temp = *head;
        (*head) = (*head)->next;
    }
}
```

OUTPUT

```
PS C:\Users\amar\Desktop\data> cd "c:\Users\amar\Desktop\data\" ; if ($?) { gcc 5a.c -o 5a } ; if ($?) { .\5a }
Enter data into the list
Enter number: 2
Do you wish to continue [1/0]: 1
Enter number: 3
Do you wish to continue [1/0]: 1
Enter number: 30
Do you wish to continue [1/0]: 0

Displaying the nodes in the list:
2       3       30
Displaying the list in reverse:
30      3       2
PS C:\Users\amar\Desktop\data>
```

NAME = DEEPAK PRAKASH

ROLL NO. =2019UCS2018