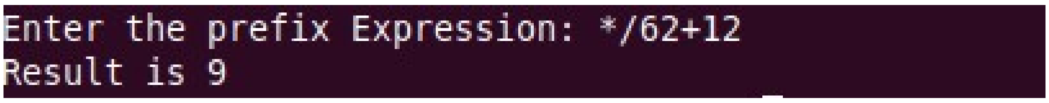
Week 4

Question 1:

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

int steck[10];
int top = -1;

void push(int value)
{

    steck[top++] = value;
}

int pop()
{

    return (steck[top--]);
}

int ope(char c)
{

    if (c == '+' || c == '-' || c == '*' || c == '/')

        return 1;

    else

        return 0;
}

void main()
{

    char prefix[10];
    int len, val, i, opr1, opr2, res;
    printf("Enter the prefix Expression: ");
    scanf("%s", prefix);
    len = strlen(prefix);
    for (i = len - 1; i >= 0; i--)
    {

        switch (ope(prefix[i]))
        {

        case 0:

            val = prefix[i];
            push(val);
            break;

        case 1:
            opr1 = pop();
            opr2 = pop();
            switch (prefix[i])
            {

            case '+':
                res = opr1 + opr2;
                break;

            case '-':
                res = opr1 - opr2;
                break;

            case '*':
                res = opr1 * opr2;
                break;

            case '/':
                res = opr1 / opr2;
                break;
            }

            push(res);
        }
    }

    printf("Result is %d\n", steck[top]);

    getchar();
}
```

Output:



```
Enter the prefix Expression: */62+12
Result is 9
```

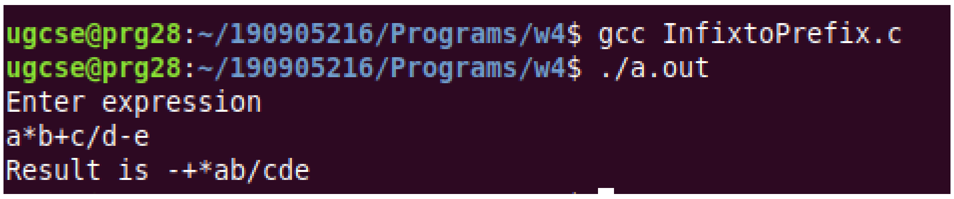
Question 2:

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

# define SIZE 1000
# define UNDER '\0'

void push (char *s, char e, int *top)
{
        if (*top == SIZE-1)
                return;
        (*top) += 1;
        *(s+(*top)) = e;
}

char pop (char *s, int *top)
{
        if (*top == -1)
                return UNDER;
        return *(s+((*top)--));
}

void reverse (char *s, int top)
{
        int i;
        for (i=0; i<=top/2; i++)
        {
                char c = *(s + i);
                *(s + i) = *(s + top - i);
                *(s + top - i) = c;
        }
}

int indexOf (char  c, char *s)
{
        char *p = strchr(s, c);
        if (p)
                return (int) (p-s);
        return -1;
}

int isOp (char op)
{
        if (indexOf(op, "+-*/") != -1)
                return 1;
        return 0;
}

int prec (char op)
```

```
{
        if (indexOf(op, ")]}") != -1)
                return 0;
        else if (indexOf(op, "+-") != -1)
                return 1;
        else if (indexOf(op, "*/") != -1)
                return 2;
        return -1;
}

char *toPrefixExpn (char *e)
{
        int top1, top2;
        top1 = -1;
        top2 = -1;
        char *pre = (char *) calloc (SIZE, sizeof(char));
        char *op = (char *) calloc (SIZE, sizeof(char));
        int l, i;
        l = strlen(e);
        for (i=l-1; i>=0; --i)
        {
                char z = *(e+i);
                if ((isdigit(z) || isalpha(z)) == 1)
                        push (pre, z, &top1);
                else if (prec(z) == 0)
                        push (op, z, &top2);
                else if (isOp(z) == 1)
                {
                        while ((top2 != -1) && prec(z) < prec(*(op+top2)))
                        {
                                char o = pop (op, &top2);
                                if (isOp(o) == 1)
                                        push (pre, o, &top1);
                        }
                        push (op, z, &top2);
                }
                else if (indexOf (z, "([{") != -1)
                {
                        while (prec(*(op + top2)) != 0)
                                push (pre, pop (op, &top2), &top1);
                        pop (op, &top2);
                }
                else
                        continue;
        }
        while (top2 != -1)
                push (pre, pop (op, &top2), &top1);
        reverse (pre, top1);
        return pre;
}

int main()
```

```
{
        char *expn = (char *) calloc (SIZE, sizeof(char));
        printf ("Enter expression\n");
        scanf ("%s", expn);
        char *p = toPrefixExpn (expn);
        printf ("Result is %s\n", p);
        return 0;
}
```

Output:



```
ugcse@prg28:~/190905216/Programs/w4$ gcc InfixtoPrefix.c
ugcse@prg28:~/190905216/Programs/w4$ ./a.out
Enter expression
a*b+c/d-e
Result is -+*ab/cde
```

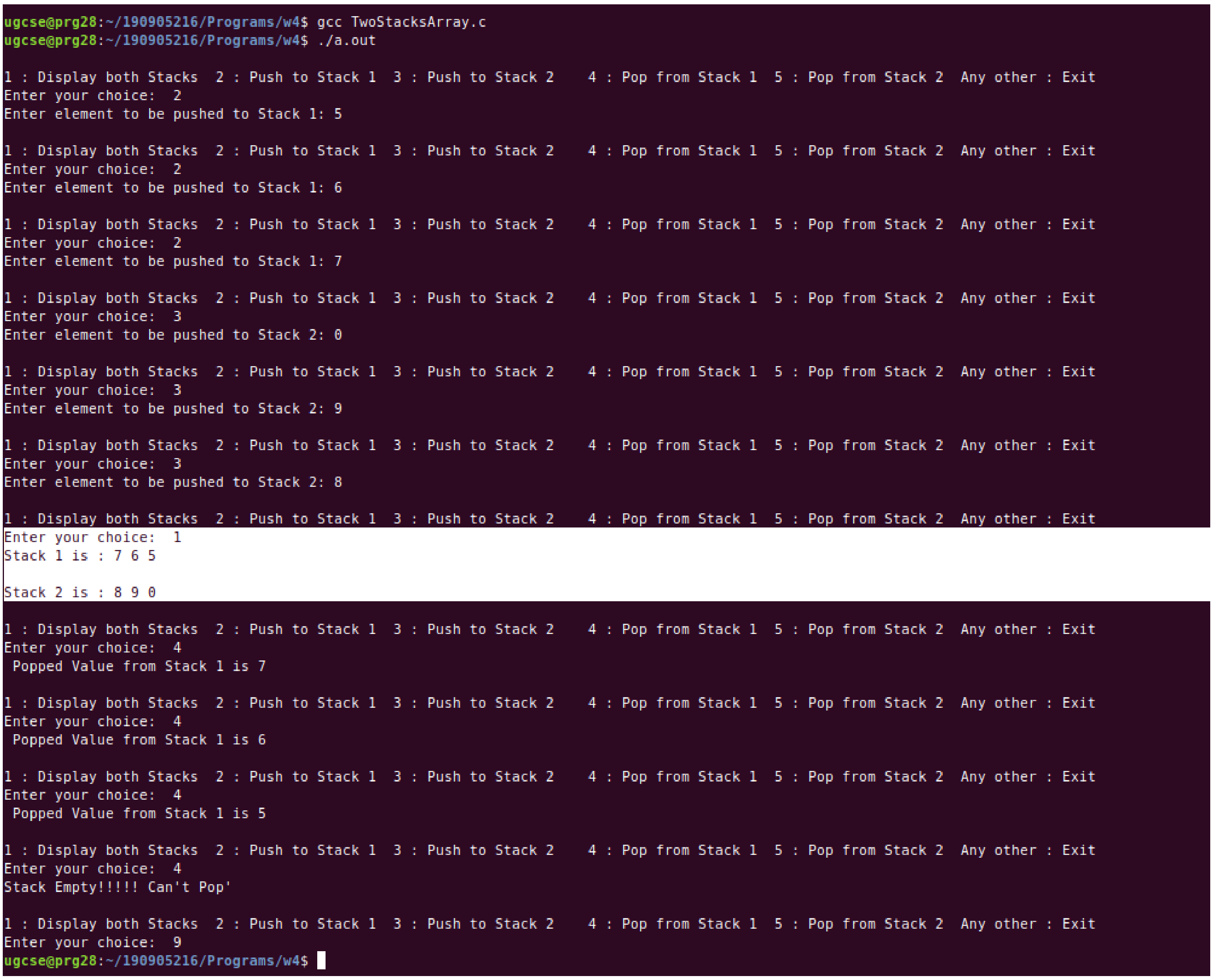
PTO->

Question 3:

```
#include <stdio.h>
#define SIZE 10

int ar[SIZE];
int top1 = -1;
int top2 = SIZE;
//Opposite ends of the Array

void push_stack1(int data) {

  if (top1 < top2 - 1)
    ar[++top1] = data;
  else
    printf("Stack Full! Cannot Push\n");
}

void push_stack2(int data) {

  if (top1 < top2 - 1)
    ar[--top2] = data;
  else
    printf("Stack Full! Cannot Push\n");
}

void pop_stack1() {
  if (top1 >= 0)
  {
    int popped_value = ar[top1--];
    printf(" Popped Value from Stack 1 is %d\n", popped_value);
  }
  else
    printf("Stack Empty!!!!! Can't Pop'\n");
}

void pop_stack2() {
  if (top2 < SIZE)
  {
    int popped_value = ar[top2++];
    printf("Popped Value from Stack 2 is %d\n", popped_value);
  }
  else
```

```
    printf("Stack Empty!!!!!! Can't Pop\n");
}

void print_stack1() {
  int i;
  for (i = top1; i >= 0; --i)
  {
    printf("%d ", ar[i]);
  }
  printf("\n");
}

void print_stack2() {
  int i;
  for (i = top2; i < SIZE; ++i)
  {
    printf("%d ", ar[i]);
  }
  printf("\n");
}

int main() {

  int ar[SIZE];
  int i, choice;
  int num_of_ele;

  do
  {
    printf("\n1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1
                                5 : Pop from Stack 2  Any other : Exit");
    printf("\nEnter your choice:  ");
    scanf("%d", &choice);
    getchar();
    switch (choice)
    {
    case 1:
    printf("Stack 1 is : ");
      print_stack1();
      printf("\nStack 2 is : ");
      print_stack2();
      break;

    case 2:
      printf("Enter element to be pushed to Stack 1: ");
      scanf("%d", &i);
      push_stack1(i);
      break;

    case 3:
      printf("Enter element to be pushed to Stack 2: ");
      scanf("%d", &i);
      push_stack2(i);
      break;

    case 4:
      pop_stack1();
      break;

    case 5:
      pop_stack2();
      break;
    }

  } while (choice < 7);
}
```

Output:



```
ugcse@prg28:~/190905216/Programs/w4$ gcc TwoStacksArray.c
ugcse@prg28:~/190905216/Programs/w4$ ./a.out

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  2
Enter element to be pushed to Stack 1: 5

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  2
Enter element to be pushed to Stack 1: 6

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  2
Enter element to be pushed to Stack 1: 7

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  3
Enter element to be pushed to Stack 2: 0

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  3
Enter element to be pushed to Stack 2: 9

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  3
Enter element to be pushed to Stack 2: 8

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  1
Stack 1 is : 7 6 5

Stack 2 is : 8 9 0

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  4
 Popped Value from Stack 1 is 7

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  4
 Popped Value from Stack 1 is 6

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  4
 Popped Value from Stack 1 is 5

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  4
Stack Empty!!!!! Can't Pop'

1 : Display both Stacks  2 : Push to Stack 1  3 : Push to Stack 2    4 : Pop from Stack 1  5 : Pop from Stack 2  Any other : Exit
Enter your choice:  9
ugcse@prg28:~/190905216/Programs/w4$
```