#include <stdio.h>

#include <stdlib.h>

#include <time.h>

struct stack

{

int size;

int \*arr;

};

int pop(struct stack \*ptr, int number)

{

// 0 1 2 3 4 5 6 7 8

// 1 2 3 4 5 6 7 8 9

int selec = 0;

int val;

while (ptr->arr[selec] != number)

{

selec++;

}

if (selec == ptr->size - 1)

{

val = ptr->arr[selec];

ptr->size--;

}

else

{

val = ptr->arr[selec];

for (int i = selec; i < ptr->size - 1; i++)

{

ptr->arr[i] = ptr->arr[i + 1];

}

ptr->size--;

}

return val;

}

int main()

{

int mode;

int key = 9;

srand(time(NULL));

char name[10];

char \*A = (char \*)malloc(5 \* sizeof(char));

char \*B = (char \*)malloc(5 \* sizeof(char));

char \*C = (char \*)malloc(5 \* sizeof(char));

printf("CROSS ZERO\n");

printf("Enter your name\n");

scanf("%s", name);

printf("Computer v/s %s\n", name);

A[0] = ' ', B[0] = ' ', C[0] = ' ';

A[1] = '|', B[1] = '|', C[1] = '|';

A[2] = ' ', B[2] = ' ', C[2] = ' ';

A[3] = '|', B[3] = '|', C[3] = '|';

A[4] = ' ', B[4] = ' ', C[4] = ' ';

// initialisation done

printf("Computer --> X, %s --> O\n", name);

printf("\n");

printf("%s\n", A);

printf("%s\n", B);

printf("%s\n", C);

printf("\n");

printf("Choose among the following to fill the respective block\n");

printf("1 2 3\n");

printf("4 5 6\n");

printf("7 8 9\n");

printf("\n");

// data fill in

struct stack \*queue = (struct stack \*)malloc(sizeof(struct stack));

queue->size = 9;

queue->arr = (int \*)malloc(queue->size \* sizeof(int));

queue->arr[0] = 1;

queue->arr[1] = 2;

queue->arr[2] = 3;

queue->arr[3] = 4;

queue->arr[4] = 5;

queue->arr[5] = 6;

queue->arr[6] = 7;

queue->arr[7] = 8;

queue->arr[8] = 9;

// stack insertion

// queue 0 1 2 3 4 5 6 7 8 9

// funtion logic creation

// 1 2 3

// 4 5 6

// 7 8 9

for (int i = 0; i < 5; i++)

{

if (A[0] == 'O' && A[2] == 'O' && A[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[0] == 'X' && A[2] == 'X' && A[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (B[0] == 'O' && B[2] == 'O' && B[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (B[0] == 'X' && B[2] == 'X' && B[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (C[0] == 'O' && C[2] == 'O' && C[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (C[0] == 'X' && C[2] == 'X' && C[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[0] == 'O' && B[0] == 'O' && C[0] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[0] == 'X' && B[0] == 'X' && C[0] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[2] == 'O' && B[2] == 'O' && C[2] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[2] == 'X' && B[2] == 'X' && C[2] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[4] == 'O' && B[4] == 'O' && C[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[4] == 'X' && B[4] == 'X' && C[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[4] == 'X' && B[2] == 'X' && C[0] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[4] == 'O' && B[2] == 'O' && C[0] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[0] == 'X' && B[2] == 'X' && C[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[0] == 'O' && B[2] == 'O' && C[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

else

{

printf("Your turn:\n");

scanf("%d", &mode);

switch (mode)

{

case 1:

A[0] = 'O';

break;

case 2:

A[2] = 'O';

break;

case 3:

A[4] = 'O';

break;

case 4:

B[0] = 'O';

break;

case 5:

B[2] = 'O';

break;

case 6:

B[4] = 'O';

break;

case 7:

C[0] = 'O';

break;

case 8:

C[2] = 'O';

break;

case 9:

C[4] = 'O';

break;

default:

printf("Invalid");

goto END;

break;

}

printf("\n");

printf("%s\n", A);

printf("%s\n", B);

printf("%s\n", C);

printf("\n");

if (A[0] == 'O' && A[2] == 'O' && A[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[0] == 'X' && A[2] == 'X' && A[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (B[0] == 'O' && B[2] == 'O' && B[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (B[0] == 'X' && B[2] == 'X' && B[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (C[0] == 'O' && C[2] == 'O' && C[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (C[0] == 'X' && C[2] == 'X' && C[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[0] == 'O' && B[0] == 'O' && C[0] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[0] == 'X' && B[0] == 'X' && C[0] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[2] == 'O' && B[2] == 'O' && C[2] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[2] == 'X' && B[2] == 'X' && C[2] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[4] == 'O' && B[4] == 'O' && C[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[4] == 'X' && B[4] == 'X' && C[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[4] == 'X' && B[2] == 'X' && C[0] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[4] == 'O' && B[2] == 'O' && C[0] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

if (A[0] == 'X' && B[2] == 'X' && C[4] == 'X')

{

printf("Computer won :(\n");

goto END;

}

if (A[0] == 'O' && B[2] == 'O' && C[4] == 'O')

{

printf("You won :D !!!\n");

goto END;

}

pop(queue, mode);

key--;

srand(time(NULL));

int comp = rand() % key;

// computers turn

printf("Computer's turn :\n");

switch (queue->arr[comp])

{

case 1:

A[0] = 'X';

break;

case 2:

A[2] = 'X';

break;

case 3:

A[4] = 'X';

break;

case 4:

B[0] = 'X';

break;

case 5:

B[2] = 'X';

break;

case 6:

B[4] = 'X';

break;

case 7:

C[0] = 'X';

break;

case 8:

C[2] = 'X';

break;

case 9:

C[4] = 'X';

break;

default:

printf("Invalid");

goto END;

break;

}

pop(queue, queue->arr[comp]);

key--;

printf("\n");

printf("%s\n", A);

printf("%s\n", B);

printf("%s\n", C);

printf("\n");

}

}

printf("Match tied\n");

END:

return 0;

}