LAB ASSIGNMENT 3

U24CS076

RUSHANG BAGADA

Q1:

*#include* <stdio.h>

*#define* *MAX* 5

*void* *push*(*int* stack*[]*, *int* *\**top, *int* value)

{

*if* (*\**top *==* MAX *-* 1)

    {

*printf*("Stack overflow\n");

*return*;

    }

    (*\**top)*++*;

*stack*[*\**top] *=* value;

}

*void* *pop*(*int* stack*[]*, *int* *\**top)

{

*if* (*\**top *==* *-*1)

    {

*printf*("Stack underflow\n");

*return*;

    }

    (*\**top)*--*;

}

*void* *peek*(*int* stack*[]*, *int* *\**top)

{

*if* (*\**top *==* *-*1)

    {

*printf*("Stack is empty\n");

*return*;

    }

*printf*("Top element: %d\n", *stack*[*\**top]);

}

*void* *display*(*int* stack*[]*, *int* top)

{

*if* (top *==* *-*1)

    {

*printf*("Stack is empty\n");

*return*;

    }

*printf*("Stack elements: ");

*for* (*int* i *=* 0; i *<=* top; i*++*)

    {

*printf*("%d ", *stack*[i]);

    }

*printf*("\n");

}

*void* *isempty*(*int* top)

{

*if* (top *==* *-*1)

    {

*printf*("stack is empty\n");

    }*else*{

*printf*("stack is not empty\n");

    }

}

*void* *isfull*(*int* top)

{

*if* (top *==* MAX *-*1)

    {

*printf*("stack is full\n");

    }

*else*

    {

*printf*("stack is not full\n");

    }

}

*int* *main*()

{

*int* *stack*[MAX];

*int* top *=* *-*1;

*int* choice;

*int* value;

*do*

    {

*printf*("1. Push\n");

*printf*("2. Pop\n");

*printf*("3. Peek\n");

*printf*("4. isempty\n");

*printf*("5. isfull\n");

*printf*("6. Display\n");

*printf*("7. Exit\n");

*printf*("Enter your choice: ");

*scanf*("%d", *&*choice);

*switch* (choice)

        {

*case* 1:

*printf*("Enter the value to push: ");

*scanf*("%d", *&*value);

*push*(stack, *&*top, value);

*break*;

*case* 2:

*pop*(stack, *&*top);

*break*;

*case* 3:

*peek*(stack, *&*top);

*break*;

*case* 4:

*isempty*(top);

*break*;

*case* 5:

*isfull*(top);

*break*;

*case* 6:

*display*(stack, top);

*break*;

*case* 7:

*printf*("Exiting...\n");

*break*;

*default*:

*printf*("Invalid choice\n");

*break*;

        }

    } *while* (choice *!=* 7);

*return* 0;

}

OUTPUT:

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 1

Enter the value to push: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 1

Enter the value to push: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 2

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 3

Top element: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 4

stack is not empty

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 5

stack is not full

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 6

Stack elements: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 7

Exiting...

Q2:

*#include* <stdio.h>

*#define* *MAX* 5

*void* *push*(*int* stack*[]*, *int* *\**top, *int* value)

{

*if* (*\**top *==* *-*1)

    {

*printf*("Stack overflow\n");

*return*;

    }

    stack[*\**top] *=* value;

    (*\**top)*--*; *// Move the top down*

}

*void* *pop*(*int* stack*[]*, *int* *\**top)

{

*if* (*\**top *==* *MAX* *-* 1)

    {

*printf*("Stack underflow\n");

*return*;

    }

    (*\**top)*++*; *// Move the top up*

}

*void* *peek*(*int* stack*[]*, *int* top)

{

*if* (top *==* *MAX* *-* 1)

    {

*printf*("Stack is empty\n");

*return*;

    }

*printf*("Top element: %d\n", stack[top *+* 1]);

}

*void* *display*(*int* stack*[]*, *int* top)

{

*if* (top *==* *MAX* *-* 1)

    {

*printf*("Stack is empty\n");

*return*;

    }

*printf*("Stack elements: ");

*for* (*int* *i* *=* top *+* 1; *i* *<* *MAX*; *i++*)

    {

*printf*("%d ", stack[*i*]);

    }

*printf*("\n");

}

*void* *isempty*(*int* top)

{

*if* (top *==* *MAX* *-* 1)

    {

*printf*("Stack is empty\n");

    }

*else*

    {

*printf*("Stack is not empty\n");

    }

}

*void* *isfull*(*int* top)

{

*if* (top *==* *-*1)

    {

*printf*("Stack is full\n");

    }

*else*

    {

*printf*("Stack is not full\n");

    }

}

*int* *main*()

{

*int* *stack*[*MAX*];

*int* *top* *=* *MAX* *-* 1; *// Initialize top to the last index*

*int* *choice*;

*int* *value*;

*do*

    {

*printf*("1. Push\n");

*printf*("2. Pop\n");

*printf*("3. Peek\n");

*printf*("4. isempty\n");

*printf*("5. isfull\n");

*printf*("6. Display\n");

*printf*("7. Exit\n");

*printf*("Enter your choice: ");

*scanf*("%d", *&choice*);

*switch* (*choice*)

        {

*case* 1:

*printf*("Enter the value to push: ");

*scanf*("%d", *&value*);

*push*(*stack*, *&top*, *value*);

*break*;

*case* 2:

*pop*(*stack*, *&top*);

*break*;

*case* 3:

*peek*(*stack*, *top*);

*break*;

*case* 4:

*isempty*(*top*);

*break*;

*case* 5:

*isfull*(*top*);

*break*;

*case* 6:

*display*(*stack*, *top*);

*break*;

*case* 7:

*printf*("Exiting...\n");

*break*;

*default*:

*printf*("Invalid choice\n");

*break*;

        }

    } *while* (*choice* *!=* 7);

*return* 0;

}

OUTPUT:

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 1

Enter the value to push: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 1

Enter the value to push: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 2

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 3

Top element: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 4

stack is not empty

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 5

stack is not full

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 6

Stack elements: 3

1. Push

2. Pop

3. Peek

4. isempty

5. isfull

6. Display

7. Exit

Enter your choice: 7

Exiting...

Q3:

*#include* <stdio.h>

*#include* <stdlib.h>

*#define* *MAX* 10

*typedef* *struct*

{

*int* *arr*[*MAX*];

*int* *top1*;

*int* *top2*;

} TwoStacks;

*void* *initStacks*(TwoStacks *\**stacks)

{

    stacks->*top1* *=* *-*1;

    stacks->*top2* *=* *MAX*;

}

*void* *push1*(TwoStacks *\**stacks, *int* value)

{

*if* (stacks->*top1* *+* 1 *==* stacks->*top2*)

    {

*printf*("Stack 1 overflow\n");

*return*;

    }

    stacks->*arr*[*++*stacks->*top1*] *=* value;

}

*void* *push2*(TwoStacks *\**stacks, *int* value)

{

*if* (stacks->*top2* *-* 1 *==* stacks->*top1*)

    {

*printf*("Stack 2 overflow\n");

*return*;

    }

    stacks->*arr*[*--*stacks->*top2*] *=* value;

}

*int* *pop1*(TwoStacks *\**stacks)

{

*if* (stacks->*top1* *==* *-*1)

    {

*printf*("Stack 1 underflow\n");

*return* *-*1;

    }

*return* stacks->*arr*[stacks->*top1--*];

}

*int* *pop2*(TwoStacks *\**stacks)

{

*if* (stacks->*top2* *==* *MAX*)

    {

*printf*("Stack 2 underflow\n");

*return* *-*1;

    }

*return* stacks->*arr*[stacks->*top2++*];

}

*int* *peek1*(TwoStacks *\**stacks)

{

*if* (stacks->*top1* *==* *-*1)

    {

*printf*("Stack 1 is empty\n");

*return* *-*1;

    }

*return* stacks->*arr*[stacks->*top1*];

}

*int* *peek2*(TwoStacks *\**stacks)

{

*if* (stacks->*top2* *==* *MAX*)

    {

*printf*("Stack 2 is empty\n");

*return* *-*1;

    }

*return* stacks->*arr*[stacks->*top2*];

}

*int* *isEmpty1*(TwoStacks *\**stacks)

{

*return* stacks->*top1* *==* *-*1;

}

*// Function to check if Stack 2 is empty*

*int* *isEmpty2*(TwoStacks *\**stacks)

{

*return* stacks->*top2* *==* *MAX*;

}

*// Function to display the contents of both stacks*

*void* *displayStacks*(TwoStacks *\**stacks)

{

*printf*("Stack 1: ");

*for* (*int* *i* *=* 0; *i* *<=* stacks->*top1*; *i++*)

    {

*printf*("%d ", stacks->*arr*[*i*]);

    }

*printf*("\n");

*printf*("Stack 2: ");

*for* (*int* *i* *=* *MAX* *-* 1; *i* *>=* stacks->*top2*; *i--*)

    {

*printf*("%d ", stacks->*arr*[*i*]);

    }

*printf*("\n");

}

*int* *main*()

{

    TwoStacks *stacks*;

*initStacks*(*&stacks*);

*int* *choice*, *value*;

*do*

    {

*printf*("1. Push to Stack 1\n");

*printf*("2. Push to Stack 2\n");

*printf*("3. Pop from Stack 1\n");

*printf*("4. Pop from Stack 2\n");

*printf*("5. Peek Stack 1\n");

*printf*("6. Peek Stack 2\n");

*printf*("7. Display Stacks\n");

*printf*("8. Exit\n");

*printf*("Enter your choice: ");

*scanf*("%d", *&choice*);

*switch* (*choice*)

        {

*case* 1:

*printf*("Enter value to push to Stack 1: ");

*scanf*("%d", *&value*);

*push1*(*&stacks*, *value*);

*break*;

*case* 2:

*printf*("Enter value to push to Stack 2: ");

*scanf*("%d", *&value*);

*push2*(*&stacks*, *value*);

*break*;

*case* 3:

*value* *=* *pop1*(*&stacks*);

*if* (*value* *!=* *-*1)

            {

*printf*("Popped from Stack 1: %d\n", *value*);

            }

*break*;

*case* 4:

*value* *=* *pop2*(*&stacks*);

*if* (*value* *!=* *-*1)

            {

*printf*("Popped from Stack 2: %d\n", *value*);

            }

*break*;

*case* 5:

*value* *=* *peek1*(*&stacks*);

*if* (*value* *!=* *-*1)

            {

*printf*("Top of Stack 1: %d\n", *value*);

            }

*break*;

*case* 6:

*value* *=* *peek2*(*&stacks*);

*if* (*value* *!=* *-*1)

            {

*printf*("Top of Stack 2: %d\n", *value*);

            }

*break*;

*case* 7:

*displayStacks*(*&stacks*);

*break*;

*case* 8:

*printf*("Exiting...\n");

*break*;

*default*:

*printf*("Invalid choice\n");

*break*;

        }

    } *while* (*choice* *!=* 8);

*return* 0;

}

OUTPUT:

1. Push to Stack 1

2. Push to Stack 2

3. Pop from Stack 1

4. Pop from Stack 2

5. Peek Stack 1

6. Peek Stack 2

7. Display Stacks

8. Exit

Enter your choice: 1

Enter value to push to Stack 1: 2

1. Push to Stack 1

2. Push to Stack 2

3. Pop from Stack 1

4. Pop from Stack 2

5. Peek Stack 1

6. Peek Stack 2

7. Display Stacks

8. Exit

Enter your choice: 2

Enter value to push to Stack 2: 2

1. Push to Stack 1

2. Push to Stack 2

3. Pop from Stack 1

4. Pop from Stack 2

5. Peek Stack 1

6. Peek Stack 2

7. Display Stacks

8. Exit

Enter your choice: 5

Top of Stack 1: 2

1. Push to Stack 1

2. Push to Stack 2

3. Pop from Stack 1

4. Pop from Stack 2

5. Peek Stack 1

6. Peek Stack 2

7. Display Stacks

8. Exit

Enter your choice: 6

Top of Stack 2: 2

1. Push to Stack 1

2. Push to Stack 2

3. Pop from Stack 1

4. Pop from Stack 2

5. Peek Stack 1

6. Peek Stack 2

7. Display Stacks

8. Exit

Enter your choice: 8

Exiting...

Q4:

*#include* <stdio.h>

*#include* <string.h>

*int* *main*()

{

*char* *str*[20];

*char* *stack*[20];

*int* *top* *=* *-*1;

*printf*("Enter a string: ");

*scanf*("%s", *str*);

*int* *len* *=* *strlen*(*str*);

*for* (*int* *i* *=* 0; *i* *<* *len*; *i++*)

    {

*stack*[*++top*] *=* *str*[*i*];

    }

*for* (*int* *i* *=* 0; *i* *<* *len*; *i++*)

    {

*str*[*i*] *=* *stack*[*top--*];

    }

*printf*("Reversed string: %s\n", *str*);

*return* 0;

}

OUTPUT:

Enter a string: RUSHANG

Reversed string: GNAHSUR

Q5:

*#include* <stdio.h>

*#include* <stdbool.h>

*#include* <string.h>

*bool* *is\_alpha*(*char* ch)

{

*return* (ch *>=* 'a' *&&* ch *<=* 'z') *||* (ch *>=* 'A' *&&* ch *<=* 'Z');

}

*bool* *is\_digit*(*char* ch)

{

*return* ch *>=* '0' *&&* ch *<=* '9';

}

*bool* *is\_alnum\_or\_underscore*(*char* ch)

{

*return* *is\_alpha*(ch) *||* *is\_digit*(ch) *||* ch *==* '\_';

}

*bool* *is\_valid\_multidimensional\_array*(*const* *char* *\**declaration)

{

*const* *char* *\*valid\_types[]* *=* {"int", "float", "double", "char", "long"};

*int* *num\_valid\_types* *=* *sizeof*(*valid\_types*) */* *sizeof*(*valid\_types*[0]);

*char* *buffer*[256];

*strncpy*(*buffer*, declaration, *sizeof*(*buffer*) *-* 1);

*buffer*[*sizeof*(*buffer*) *-* 1] *=* '\0';

*size\_t* *len* *=* *strlen*(*buffer*);

*if* (*len* *<* 2 *||* *buffer*[*len* *-* 1] *!=* ';')

    {

*return* *false*;

    }

*buffer*[*len* *-* 1] *=* '\0';

*char* *\*token* *=* *strtok*(*buffer*, " ");

*bool* *valid\_type* *=* *false*;

*for* (*int* *i* *=* 0; *i* *<* *num\_valid\_types*; *++i*)

    {

*if* (*strcmp*(*token*, *valid\_types*[*i*]) *==* 0)

        {

*valid\_type* *=* *true*;

*break*;

        }

    }

*if* (*!valid\_type*)

    {

*return* *false*;

    }

*token* *=* *strtok*(*NULL*, "[");

*if* (*token* *==* *NULL* *||* *!is\_alpha*(*token*[0]))

    {

*return* *false*;

    }

*for* (*int* *i* *=* 1; *token*[*i*] *!=* '\0'; *++i*)

    {

*if* (*!is\_alnum\_or\_underscore*(*token*[*i*]))

        {

*return* *false*;

        }

    }

*while* ((*token* *=* *strtok*(*NULL*, "]")) *!=* *NULL*)

    {

*if* (*token*[0] *!=* '[')

        {

*return* *false*;

        }

*for* (*int* *i* *=* 1; *token*[*i*] *!=* '\0'; *++i*)

        {

*if* (*!is\_digit*(*token*[*i*]))

            {

*return* *false*;

            }

        }

    }

*return* *true*;

}

*int* *main*()

{

*char* *user\_input*[256];

*printf*("Enter a multi-dimensional array declaration: ");

*if* (*fgets*(*user\_input*, *sizeof*(*user\_input*), *stdin*) *!=* *NULL*)

    {

*// Remove trailing newline*

*size\_t* *len* *=* *strlen*(*user\_input*);

*if* (*len* *>* 0 *&&* *user\_input*[*len* *-* 1] *==* '\n')

        {

*user\_input*[*len* *-* 1] *=* '\0';

        }

*if* (*is\_valid\_multidimensional\_array*(*user\_input*))

        {

*printf*("Valid array declaration.\n");

        }

*else*

        {

*printf*("Invalid array declaration.\n");

        }

    }

*return* 0;

}

OUTPUT:

Enter a multi-dimensional array declaration: int a[5][6]7[]

Invalid array declaration.

Q6:

*#include* <stdio.h>

*int* *fact*(*int* n)

{

*if* (n *==* 0)

    {

*return* 1;

    }

*return* n *\** *fact*(n *-* 1);

}

*int* *main*()

{

*int* *n*;

*printf*("Enter a number: ");

*scanf*("%d", *&n*);

*printf*("Factorial of %d is %d\n", *n*, *fact*(*n*));

*return* 0;

}

OUTPUT:

Enter a number: 6

Factorial of 6 is 720

Q7:

*#include* <stdio.h>

*// Function to calculate the nth Fibonacci number using recursion*

*int* *fibonacci*(*int* n)

{

*if* (n *<=* 0)

    {

*return* 0;

    }

*else* *if* (n *==* 1)

    {

*return* 1;

    }

*else*

    {

*return* *fibonacci*(n *-* 1) *+* *fibonacci*(n *-* 2);

    }

}

*void* *displayFibonacciSeries*(*int* N)

{

*for* (*int* *i* *=* 0; *i* *<* N; *i++*)

    {

*printf*("%d ", *fibonacci*(*i*));

    }

}

*int* *main*()

{

*int* *N*;

*printf*("Enter the number of terms: ");

*scanf*("%d", *&N*);

*printf*("First %d Fibonacci numbers are:\n", *N*);

*displayFibonacciSeries*(*N*);

*return* 0;

}

OUTPUT:

Enter the number of terms:

5

First 5 Fibonacci numbers are:

0 1 1 2 3

Q8:

STACK:

*#include* <stdbool.h> *// For boolean type*

*#include* <stdio.h>

*// Define the maximum size of the stack*

*#define* *STACK\_MAX\_SIZE* 100

*// Define the stack structure*

*typedef* *struct*

{

*int* *data*[*STACK\_MAX\_SIZE*]; *// Array to hold stack elements*

*int* *top*; *// Index of the top element*

} Stack;

*// Function to initialize the stack*

*void* *stack\_init*(Stack *\**stack)

{

    stack->*top* *=* *-*1;

}

*// Function to check if the stack is empty*

*int* *stack\_is\_empty*(*const* Stack *\**stack)

{

*return* stack->*top* *==* *-*1;

}

*// Function to check if the stack is full*

*int* *stack\_is\_full*(*const* Stack *\**stack)

{

*return* stack->*top* *==* *STACK\_MAX\_SIZE* *-* 1;

}

*// Function to push an element onto the stack*

*int* *stack\_push*(Stack *\**stack, *int* value)

{

*if* (*stack\_is\_full*(stack))

    {

*return* *false*; *// Stack is full, push operation fails*

    }

*return* stack->*data*[*++*(stack->*top*)] *=* value;

}

*// Function to pop an element from the stack*

*int* *stack\_pop*(Stack *\**stack, *int* *\**value)

{

*if* (*stack\_is\_empty*(stack))

    {

*return* *false*; *// Stack is empty, pop operation fails*

    }

*return* *\**value *=* stack->*data*[(stack->*top*)*--*];

}

*// Function to get the top element of the stack without removing it*

*int* *stack\_peek*(*const* Stack *\**stack, *int* *\**value)

{

*if* (*stack\_is\_empty*(stack))

    {

*return* *false*; *// Stack is empty, peek operation fails*

    }

*return* *\**value *=* stack->*data*[stack->*top*];

}

MAIN FILE:

*#include* <stdio.h>

*#include* "stack.h" *// Include the header file*

*int* *main*()

{

    Stack *stack*;

*int* *value*;

*// Initialize the stack*

*stack\_init*(*&stack*);

*// Push elements onto the stack*

*stack\_push*(*&stack*, 10);

*stack\_peek*(*&stack*, *&value*);

*stack\_push*(*&stack*, 20);

*stack\_peek*(*&stack*, *&value*);

*stack\_push*(*&stack*, 30);

*// Peek the top element*

*stack\_peek*(*&stack*, *&value*);

*printf*("Top element: %d\n", *value*);

*return* 0;

}

OUTPUT:  
Top element: 30