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 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

To facilitate a 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 MAX 100

// Stack using array

typedef struct stack

{

char \*arr[MAX];

int top;

} stack;

void init(stack \*s)

{

s->top = -1;

}

int is\_empty(stack \*s)

{

return s->top == -1;

}

int is\_full(stack \*s)

{

return s->top == MAX - 1;

}

void push(stack \*s, char \*str)

{

if (is\_full(s))

{

printf("Stack overflow");

exit(1);

}

s->arr[++s->top] = str;

}

char \*pop(stack \*s)

{

if (is\_empty(s))

{

printf("Stack underflow");

exit(1);

}

return s->arr[s->top--];

}

char \*peek(stack \*s)

{

if (is\_empty(s))

{

printf("Stack underflow");

exit(1);

}

return s->arr[s->top];

}

void display(stack \*s)

{

if (is\_empty(s))

{

printf("Stack underflow");

exit(1);

}

for (int i = s->top; i >= 0; i--)

{

printf("%s ", s->arr[i]);

}

printf(" ");

}

int main()

{

stack s;

init(&s);

int choice;

char \*str;

while (1)

{

printf("\

\n1. Visit a new page\

\n2. Back\

\n3. Forward\

\n4. Quit");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

if (!is\_empty(&s))

{

printf("Error: You must go back to visit a new page");

exit(1);

}

printf("Enter the page name: ");

scanf("%s", str);

push(&s, str);

break;

case 2:

if (is\_empty(&s))

{

printf("Error: No pages in back history list");

exit(1);

}

printf("You are visiting %s", peek(&s));

break;

case 3:

if (is\_empty(&s))

{

printf("Error: No pages in forward history list");

exit(1);

}

printf("You are visiting %s", pop(&s));

break;

case 4:

if (!is\_empty(&s))

{

printf("Error: You must go back to quit");

exit(1);

}

exit(0);

default:

printf("Invalid choice");

exit(1);

}

}

return 0;

}

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 facing the left direction and other two stacks are facing in the right direction.

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

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.

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.

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.

Write a program for Binary Search Tree to implement following operations:

Insertion

Deletion

Delete node with only child

Delete node with both children

Finding an element

Finding Min element

Finding Max element

Left child of the given node

Right child of the given node  
h. Finding the number of nodes, leaves nodes, full nodes, ancestors, descendants.