University of Guelph
School of Computer Science

CIS*2520: Data Structures
Fall 2024, Lab 4: Stacks and Queues
Week of Oct. 07 – 11

---

1 Stacks

A stack is an ordered collection of items. Items are inserted at the top of the stack and removed from the top as well. A stack is a dynamic object, due to its constantly changing nature. A stack is a dynamic constantly changing object.

There are 2 basic operations associated with stack:

1. "Push" is the term used to insert an element into a stack.
2. "Pop" is the term used to delete an element from the stack.

One of the ways to implement a stack in "C" is using a structure in the following way:

```
struct stack {
  int top;
  int items[STACKSIZE];
};
```

Where top is the location of the last element added to the stack. If the stack is empty this value is set to −1. The array `items[]` contains the elements of the stack.

To "`push`" an element onto the stack, we need to increment top while "`popping`" an element from the stack decrements top.

The following is an example program for manipulating a stack.

```
/*****************************************************************
                    S T A C K   I M P L E M E N T A T I O N
*****************************************************************/
#include <stdio.h>
#include <stdlib.h>
#define STACKSIZE 5
```

```
struct stack{
     int top;
     int items[STACKSIZE];
};

int empty(struct stack *ps);
void push(struct stack *ps, int x);
int pop(struct stack *ps);

int main() {
     int x, w, i;
     struct stack s;
     s.top = -1;

     /*push items into the stack*/
     printf("Input 5 Items for stack: \n");

     for (i=0; i<STACKSIZE; i++) {
          scanf("%d", &x);
          push(&s, x);
     } // end for

     /*print the items of the stack*/

     printf("\nPopping & Printing 5 Items of stack: \n");

     for (i=0; i<STACKSIZE; i++) {
          w = pop(&s);
          printf("%d \t", w);
     } // end for

     /*checkifstackisempty*/

     if (empty(&s))
          printf("\n stack is empty \n");
     else
          printf("\n stack is not empty \n");

     return 0;

} // end main

int empty(struct stack *ps) {
     return(ps->top==-1);
}

void push(struct stack *ps, int x) {
     if (ps->top == STACKSIZE-1) {
```

```
          printf("Stack overflow /n");
          exit(1);
     }
     else {
          ps->items[++(ps->top)]=x;
     }
}

int pop(struct stack *ps) {
     if(empty(ps)){
          printf("Stack underflow /n");
          exit(1);
     }
     return(ps->items[(ps->top)--]);
}
```

Play with the program, and modify the program, to learn more about using stacks.

2 Queues

---

1. Run the following snippet, and select the correct output from the options below:

```
#include <stdio.h>


int insert( int queue[], int end, int insertItem) {
    end += 1;
    queue[end] = insertItem;
    return end;
}

void show(int queue[], int front, int end) {
    for (int i = front; i <= end; i++) {
    printf("%d ", queue[i]);
    }
    printf("\n");
}

int main() {
    int queue[10];
```

```
    int item_one = 2;
    int item_two = 1;
    int item_three = 3;

    int front = 0;
    int end = - 1;

    end = insert(queue, end, item_one);
    end = insert(queue, end, item_two);
    end = insert(queue, end, item_three);

    show(queue, front, end);

    return 0;
}
```

A. 1 2 3

B. 3 2 1

C. 2 1 3

D. 3 1 2

2. What is a queue?
   A. a dictionary
   B. an array that can be modified by removing the first item and adding to the end
   C. a string
   D. None of these.

3. Which of the following code snippets is the correct function to remove the first item from the queue?

(a)

```
void remove() {
     queue[front] = -1;
}
```

(b)

```
void remove(int index) {

     front = index;
}
```

(c)

```
void remove() {
     queue[front] = null;
}
```

(d)

```
void remove() {
     front += 1;
}
```

(e) None of these

4. Assume we have a queue with a maximum size of 10, write code to add 5 structures (can be a struct of your choice) to the queue.
5. Assume we have a queue with 5 items, write code to remove the first two items in the queue.
6. Assume we have a queue with a maximum size of 10. Write check statements in your enqueue function, try and add 11 items to the queue, make sure your code prints an error.
7. Assume we have a queue with 1 item in it. Write check statements in your dequeue function, try and remove 2 items from the queue, make sure your code prints an error.