Computing Machinery I
Assignment 5

Part A: Global Variables and Separate Compilation

A FIFO queue data structure can be implemented using an array, as shown in the following C program:

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

#define QUEUESIZE   8
#define MODMASK     0x7
#define FALSE       0
#define TRUE        1

/* Function Prototypes */
void enqueue(int value);
int dequeue();
int queueFull();
int queueEmpty();
void display();

/* Global Variables */
int queue[QUEUESIZE];
int head = -1;
int tail = -1;

int main()
{
  int operation, value;

  do {
    system("clear");
    printf("### Queue Operations ###\n\n");
    printf("Press 1 - Enqueue, 2 - Dequeue, 3 - Display, 4 - Exit\n");
    printf("Your option? ");
    scanf("%d", &operation);

    switch (operation) {
    case 1:
      printf("\nEnter the positive integer value to be enqueued: ");
      scanf("%d", &value);
      enqueue(value);
      break;
    case 2:
      value = dequeue();
      if (value != -1)
       printf("\nDequeued value is %d\n", value);
      break;
    case 3:
      display();
      break;
    case 4:
      printf("\nTerminating program\n");
      exit(0);
    default:
      printf("\nInvalid option! Try again.\n");
      break;
    }
    printf("\nPress the return key to continue . . . ");
    getchar();
    getchar();
  } while (operation != 4);
```

```
  return 0;
}

void enqueue(int value)
{
  if (queueFull()) {
    printf("\nQueue overflow! Cannot enqueue into a full queue.\n");
    return;
  }

  if (queueEmpty()) {
    head = tail = 0;
  } else {
    tail = ++tail & MODMASK;
  }
  queue[tail] = value;
}

int dequeue()
{
  register int value;

  if (queueEmpty()) {
    printf("\nQueue underflow! Cannot dequeue from an empty queue.\n");
    return (-1);
  }

  value = queue[head];
  if (head == tail) {
    head = tail = -1;
  } else {
    head = ++head & MODMASK;
  }
  return value;
}

int queueFull()
{
  if (((tail + 1) & MODMASK) == head)
    return TRUE;
  else
    return FALSE;
}

int queueEmpty()
{
  if (head == -1)
    return TRUE;
  else
    return FALSE;
}

void display()
{
  register int i, j, count;

  if (queueEmpty()) {
    printf("\nEmpty queue\n");
    return;
  }

  count = tail - head + 1;
  if (count <= 0)
    count += QUEUESIZE;

  printf("\nCurrent queue contents:\n");
```

```
  i = head;
  for (j = 0; j < count; j++) {
    printf("  %d", queue[i]);
    if (i == head) {
      printf(" <-- head of queue");
    }
    if (i == tail) {
      printf(" <-- tail of queue");
    }
    printf("\n");
    i = ++i & MODMASK;
  }
}
```

Translate all functions except main() into ARMv8 assembly language, and put them into a separate assembly source code file called *a5a.asm*. These functions will be called from the main() function given above, which will be in its own C source code file called *a5aMain.c*. Also move the global variables into *a5a.asm*. Your assembly functions will call the printf() library routine. Be sure to handle the global variables and format strings in the appropriate way. Input will come from standard input. Run the program to show that it is working as expected, capturing its output using the *script* UNIX command, and name the output file *script1.txt.*

Part B: External Pointer Arrays and Command-Line Arguments

Given the following declarations in C:

```
char *month[] = {"January", "February", "March", "April", "May",
                 "June", "July", "August", "September", "October",
                 "November", "December"};
char *season[] = {"Winter", "Spring", "Summer", "Fall"};
```

create an ARMv8 assembly language program to accept as command line arguments two strings representing a date in the format *mm dd*. Your program will print the name of month, the day (with the appropriate suffix), and the season for this date. For example:

```
./a5b 12 25
December 25th is Winter
```

Be sure to use the proper suffix for the day of the month. For example, one should distinguish the 11th from the 1st, 21st, and 31st. Your program should exit, printing this error message, if the user does not supply two command-line arguments:

```
usage: a5b mm dd
```

You will need to call atoi() to convert strings to numbers, and printf() to produce the output. Be sure to do range checking for the day and month. Assume that Winter ranges from December 21 to March 20, Spring from March 21 to June 20, Summer from June 21 to September 20, and Fall from September 21 to December 20. Name your source code file *a5b.asm*. Run your program three times with different input to illustrate that it works; capture the output using the *script* UNIX command. Name the output file *script2.txt*.

New Skills need for this Assignment:

- Understanding and use of external variables in assembly
- Separate compilation
- Calling assembly functions from main()
- Calling library functions from assembly routines
- External arrays of pointers
- Command line arguments

Submit the following:

1. Your source code and 2 scripts via electronic submission. Use the *Assignment 5* Dropbox Folder in D2L to submit electronically. Your TA will assemble and run your programs to test them. Name your files *a5aMain.c* and *a5a.asm* for Part A, and *a5b.asm* for Part B, and the scripts *as script1.txt* and *script2.txt.*

Computing Machinery I
Assignment 5 Grading

Student:_____

Part A:			
	Correct use of external variable(s)	4	_____
	enqueue() function in assembly	4	_____
	dequeue() function in assembly	4	_____
	queueFull() function in assembly	4	_____
	queueEmpty() function in assembly	4	_____
	display() function in assembly	8	_____
	Linking of separate source code modules	2	_____
	Correct manipulation of queue	4	_____
Part B:			
	Command line arguments	4	_____
	Correct use of external pointer arrays	4	_____
	Calls to library functions	2	_____
	Range checking	3	_____
	Correct output	4	_____
2 Scripts showing I/O		4	_____
Complete documentation and commenting		4	_____
Design quality		4	_____
Total		**63**	_____ _____%