CS 2263 - FR01A

Lab 7

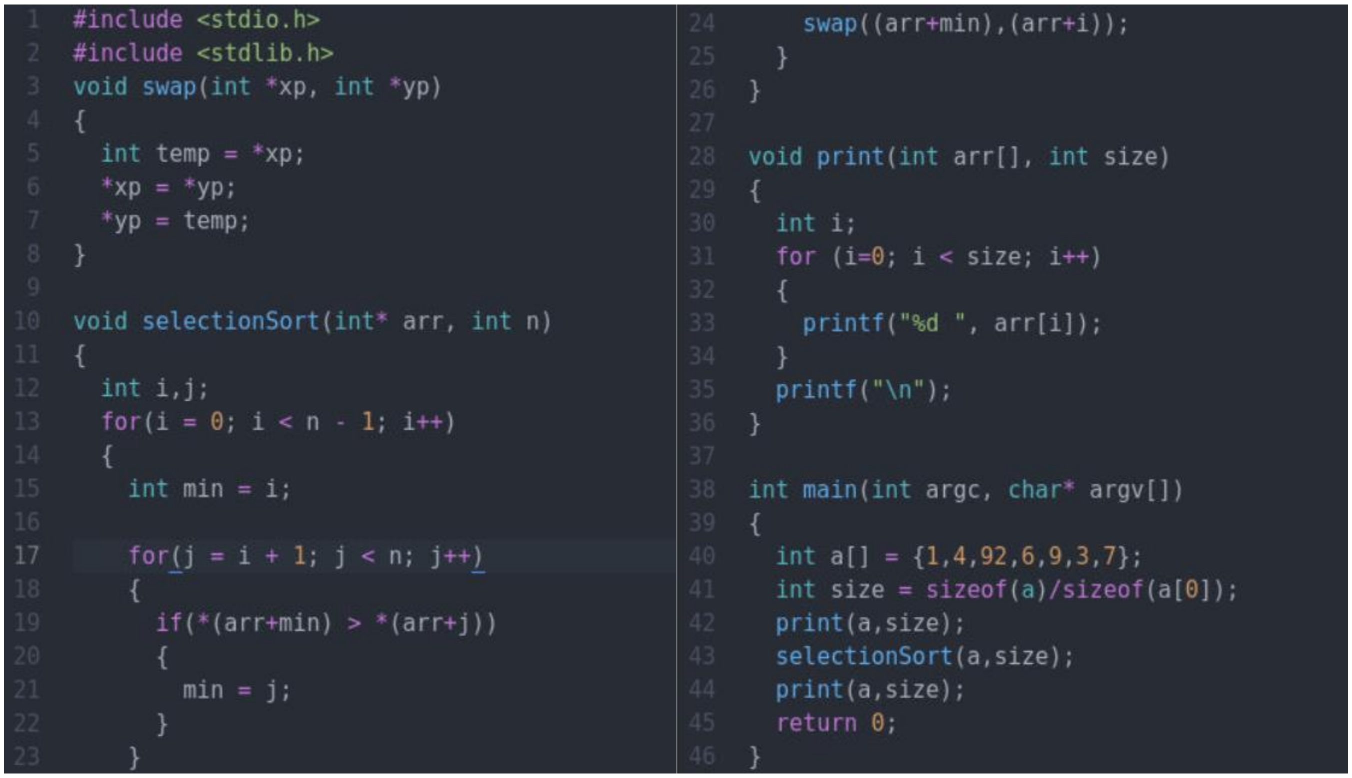
By Ngoc Phuong Anh Nguyen - 3712361

December 2021

Question 1:

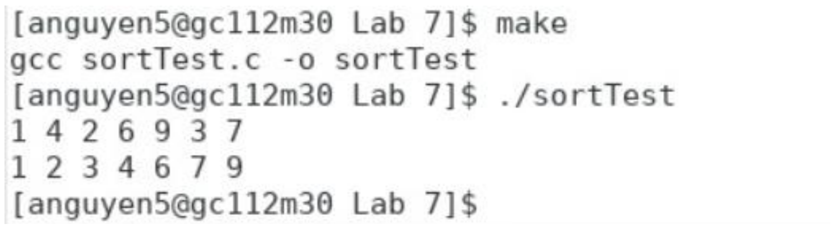
Create a standalone function (not associated with a module) that sorts using your favourite sorting algorithm (I know that you have one!) that isn't qsort() from C's stdlib. Test it using a stack-declared array of integers in a simple test program.

```
$ sortTest
```



```c
#include <stdio.h>
#include <stdlib.h>
void swap(int *xp, int *yp)
{
  int temp = *xp;
  *xp = *yp;
  *yp = temp;
}

void selectionSort(int* arr, int n)
{
  int i,j;
  for(i = 0; i < n - 1; i++)
  {
    int min = i;

    for(j = i + 1; j < n; j++)
    {
      if(*(arr+min) > *(arr+j))
      {
        min = j;
      }
    }
    swap((arr+min),(arr+i));
  }
}

void print(int arr[], int size)
{
  int i;
  for (i=0; i < size; i++)
  {
    printf("%d ", arr[i]);
  }
  printf("\n");
}

int main(int argc, char* argv[])
{
  int a[] = {1,4,92,6,9,3,7};
  int size = sizeof(a)/sizeof(a[0]);
  print(a,size);
  selectionSort(a,size);
  print(a,size);
  return 0;
}
```

Figure 1: Source Code Of Question 1

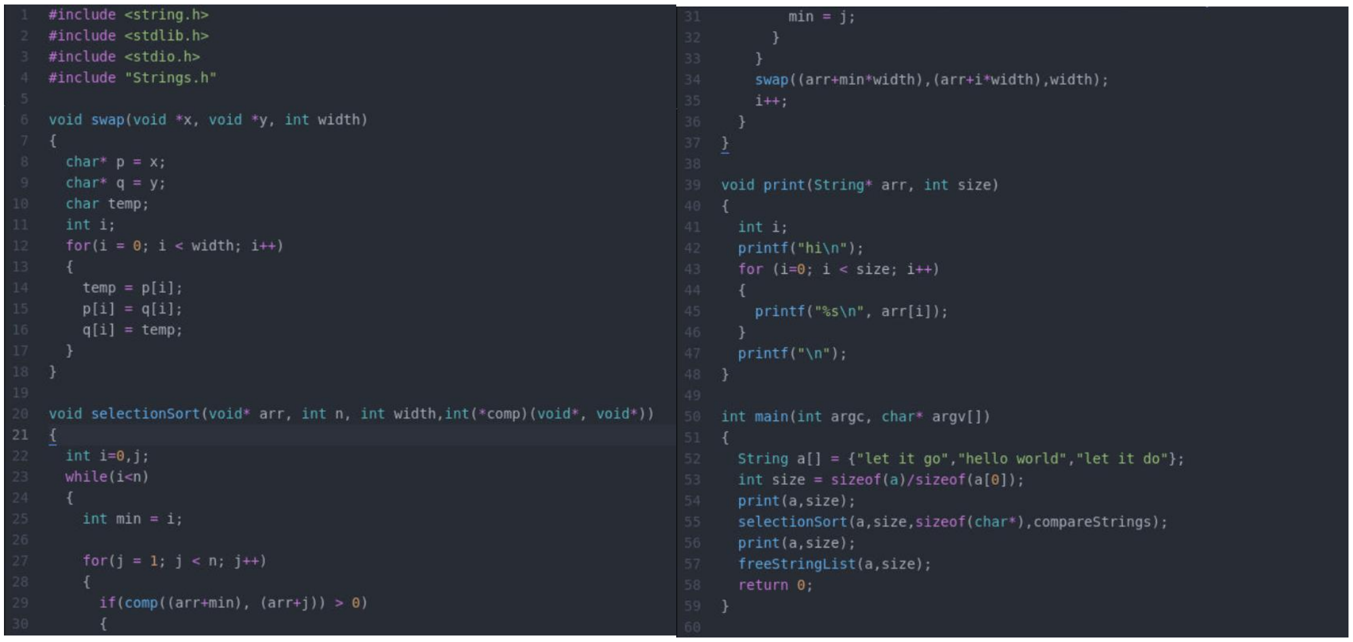


```
[anguyen5@gc112m30 Lab 7]$ make
gcc sortTest.c -o sortTest
[anguyen5@gc112m30 Lab 7]$ ./sortTest
1 4 2 6 9 3 7
1 2 3 4 6 7 9
[anguyen5@gc112m30 Lab 7]$
```

Figure 2: Make command output and output result of question 1

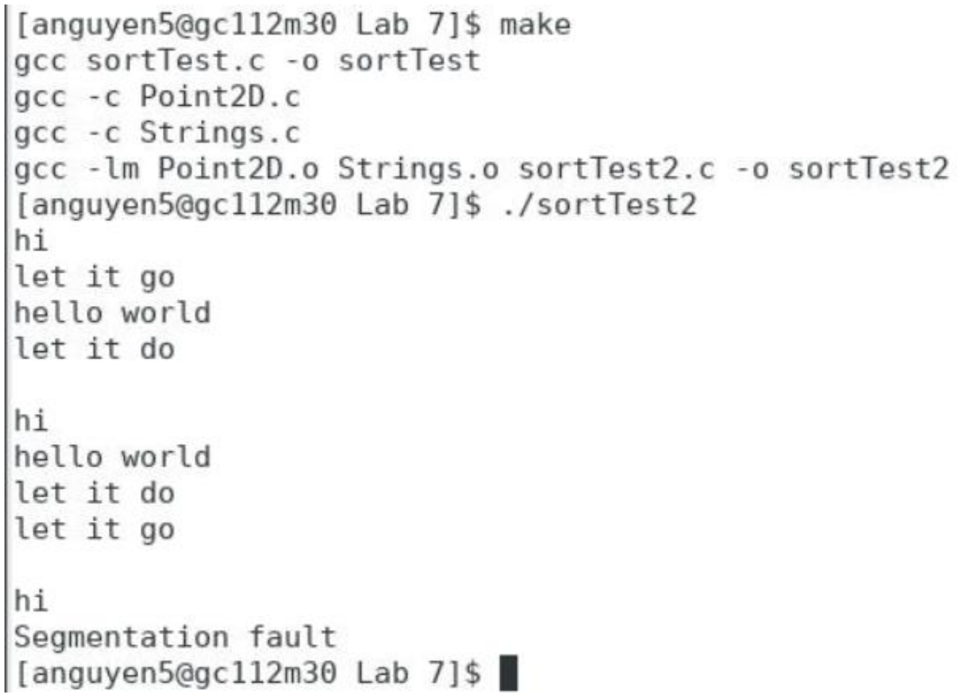
Question 2:

Modify your sorting function so that, like qsort(), you can pass a pointer to a comparison function as a parameter. You will need to do some online research to discover the technique to do this. Searching for C pointers to functions should do the trick. Using your program from Lab5 Exercise 4 (stringListSortTest), call your sorting function instead.



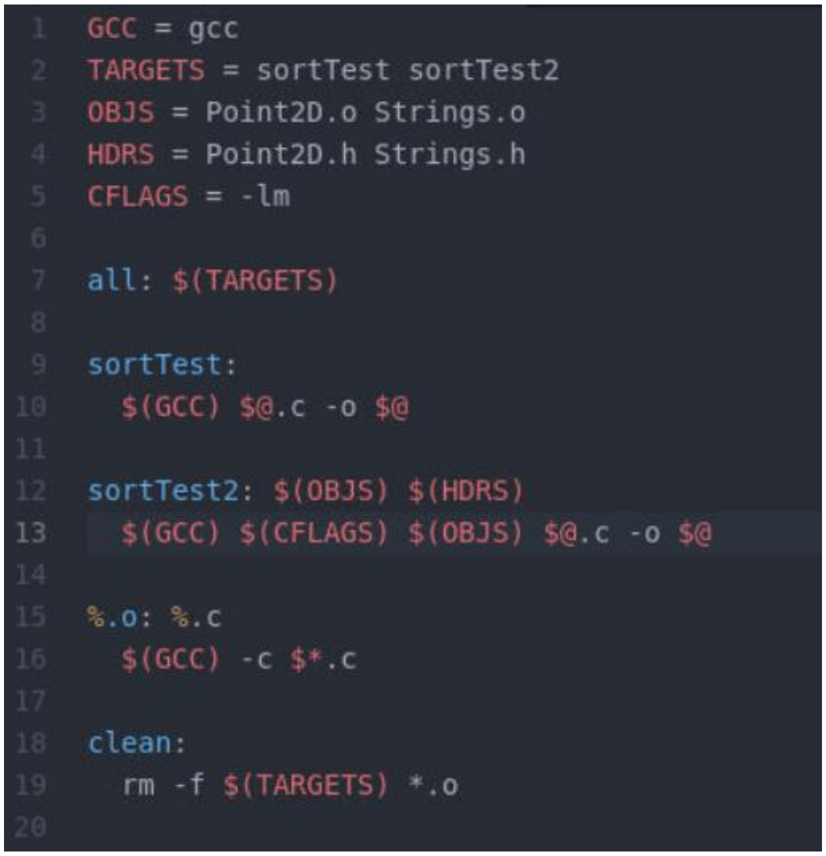
```c
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include "Strings.h"

void swap(void *x, void *y, int width)
{
  char* p = x;
  char* q = y;
  char temp;
  int i;
  for(i = 0; i < width; i++)
  {
    temp = p[i];
    p[i] = q[i];
    q[i] = temp;
  }
}

void selectionSort(void* arr, int n, int width,int(*comp)(void*, void*))
{
  int i=0,j;
  while(i<n)
  {
    int min = i;

    for(j = 1; j < n; j++)
    {
      if(comp((arr+min), (arr+j)) > 0)
      {
        min = j;
      }
    }
    swap((arr+min*width),(arr+i*width),width);
    i++;
  }
}

void print(String* arr, int size)
{
  int i;
  printf("hi\n");
  for (i=0; i < size; i++)
  {
    printf("%s\n", arr[i]);
  }
  printf("\n");
}

int main(int argc, char* argv[])
{
  String a[] = {"let it go","hello world","let it do"};
  int size = sizeof(a)/sizeof(a[0]);
  print(a,size);
  selectionSort(a,size,sizeof(char*),compareStrings);
  print(a,size);
  freeStringList(a,size);
  return 0;
}
```

Figure 3: Source Code Of Question 2



```
[anguyen5@gc112m30 Lab 7]$ make
gcc sortTest.c -o sortTest
gcc -c Point2D.c
gcc -c Strings.c
gcc -lm Point2D.o Strings.o sortTest2.c -o sortTest2
[anguyen5@gc112m30 Lab 7]$ ./sortTest2
hi
let it go
hello world
let it do

hi
hello world
let it do
let it go

hi
Segmentation fault
[anguyen5@gc112m30 Lab 7]$ 
```

Figure 4: Make command output and output result of question 2



```
GCC = gcc
TARGETS = sortTest sortTest2
OBJS = Point2D.o Strings.o
HDRS = Point2D.h Strings.h
CFLAGS = -lm

all: $(TARGETS)

sortTest:
  $(GCC) $@.c -o $@

sortTest2: $(OBJS) $(HDRS)
  $(GCC) $(CFLAGS) $(OBJS) $@.c -o $@

%.o: %.c
  $(GCC) -c $*.c

clean:
  rm -f $(TARGETS) *.o

```

Figure 5: makefile