Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

Faculty of Science, Technology, Engineering and Mathematics

School of Computer Science & Statistics

Integrated Computer Science **Michaelmas Term**
Computer Science (Joint Honours)
Computer Science, Linguistics and Langauge
Junior Freshman

CSU11021 – Introduction to Computing I

Assignment Project Exam Help

Thursday, 15 December 2022 **ONLINE** **15:00 – 18:00**

https://tutorcs.com

Dr Jonathan Dukes

WeChat: cstutorcs

Instructions to Candidates

Attempt ALL parts.

The total number of marks is 100.

This is an individual assessment. Tools similar to *TurnItIn* will be used to measure the similarity of solutions. Provide references for any sources you use to develop your solution.

You must not communicate with anyone in relation to the examination either during the examination or for 1 hour after the scheduled end time of the examination.

Submit a completed declaration on Blackboard, using the template provided, confirming that the work submitted is your own.

Submit your ARM Assembly Language program at `https://submit.scss.tcd.ie`.

You may submit your program up to eight times without penalty. Each subsequent attempt will attract a penalty of 2 marks, up to a maximum penalty of 12 marks.

© UNIVERSITY OF DUBLIN 2022

Each part of this examination is cumulative, building on the functionality of preceding parts. Correctly implementing each successive part will cause your program to pass more Submitty tests. You do not need to submit separate solutions for each part. You only need to submit your solution for the final part that you attempt. You may, if you wish, submit attempts for intermediate parts to check your solution. Submissions for intermediate parts will count towards your total of eight penalty-free attempts.

You must provide pseudocode comments to explain your approach.

The mark you receive will be based on:

(i) automated testing of your program by Submitty and [60 marks]

(ii) an evaluation of the quality of your pseudo-code comments, your use of appropriate assembly language features, your overall approach and the presentation of your program. [40 marks]

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

First, some definitions

A **"substring"** of an ASCII NULL-terminated string is a sequence of one or more characters at any position in the string. The example below highlights a substring containing the characters "**XYZ**".

"ABCD**XYZ**EFG"

A **"prefix"** of an ASCII NULL-terminated string is a substring appearing at the start of the string. The example below highlights a prefix containing the characters "**XYZ**"

"**XYZ**ABCDEFG"

© UNIVERSITY OF DUBLIN 2022

Part 1 [9 Submitty autograding marks]

Two ASCII NULL-terminated strings, *A* and *B*, are stored in Random Access Memory (RAM). Write an ARM Assembly Language program that will calculate the length of the longest prefix of string *A* that exactly matches a prefix of string *B*.

For example, given the strings *A* and *B* below, your program should give a result of 3 in `R0`. The matching prefixes have been highlighted.

string A: "**ABC**WXYZ"

string B: "**ABC**PQRST"

The start addresses of strings *A* and *B* are in registers `R1` and `R2`. Your program should store its result in register `R0`.

Assignment Project Exam Help

Part 2 [18 Submitty autograding marks]

https://tutorcs.com

Extend the functionality of your program from Part 1 to calculate the length of the longest prefix of *A* that matches a substring *anywhere* in *B*. Your program should continue to store its result in register `R0`.

WeChat: cstutorcs

For example, given the strings *A* and *B* below, your program should give a result of 4 in `R0`. The matching prefix of *A* and substring of *B* have been highlighted.

string A: "**ABCD**WXYZ"

string B: "ABCPQ**ABCD**RST"

© UNIVERSITY OF DUBLIN 2022

Part 3 [15 Submitty autograding marks]

Extend your program again to find the length of the longest substring *anywhere* in A that matches a substring *anywhere* in B. Your program should store its result in register `R0`.

For example, given the strings A and B below, your program should give a result of 5 in `R0`. The matching substrings have been highlighted.

string A: "ABCDWX**ABCDE**YZ"

string B: "ABCPQ**ABCDE**RST"

Part 4 [18 Submitty autograding marks]

Extend your program one more time to remove the longest matching substring from strings A and B. When removing the substrings from A and B, your program should overwrite the substrings with the characters that immediately follow the substrings to "fill the gap". Your program should continue to store the length of the removed substring in register `R0`.

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

For example, given the same two strings, A and B, as the example in Part 3:

string A: "ABCDWX**ABCDE**YZ"

string B: "ABCPQ**ABCDE**RST"

your program should modify the original strings A and B in memory, replacing them with the following strings:

string A: "ABCDWXYZ"

string B: "ABCPQRST"

© UNIVERSITY OF DUBLIN 2022

ARM Conditional Branch Instructions

Description	Symbol	Java	Instruction	Mnemonic
Equality				
equal	=	==	BEQ	**EQ**ual
not equal	≠	!=	BNE	**N**ot **E**qual
Inequality (unsigned values)				
less than	<	<	BLO (or BCC)	**LO**wer
less than or equal	≤	<=	BLS	**L**ower or **S**ame
greater than or equal	≥	>=	BHS (or BCS)	**H**igher or **S**ame
greater than	>	>	BHI	**HI**gher
Inequality (signed values)				
less than	<	<	BLT	**L**ess **T**han
less than or equal	≤	<=	BLE	**L**ess than or **E**qual
greater than or equal	≥	>=	BGE	**G**reater than or **E**qual
greater than	>	>	BGT	**G**reater **T**han
Flags				
Negative Set			BMI	**MI**nus
Negative Clear			BPL	**PL**us
Carry Set			BCS (or BHS)	**C**arry **S**et
Carry Clear			BCC (or BLO)	**C**arry **C**lear
Overflow Set			BVS	o**V**erflow **S**et
Overflow Clear			BVC	o**V**erflow **C**lear
Zero Set			BEQ	**EQ**ual
Zero Clear			BNE	**N**ot **E**qual

Assignment Project Exam Help
https://tutorcs.com
WeChat: cstutorcs

ASCII Table

hex	symbol	hex	symbol	hex	symbol	hex	symbol	hex	symbol	hex	symbol
20	[SPACE]	30	0	40	@	50	P	60	`	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	[DEL]

© UNIVERSITY OF DUBLIN 2022