# Algorithms Assignment

**Q3 –**

1. I used the linear search algorithm to search on the main list for all full-time students.
2. The big O of the linear search is O(n). As it goes through all the data individually. It searches for the full time modules in this case “1” (part time is 0), it searches I tin the structure in the array of “type” once it has found it prints the surname of the student enrolled in the first index, then continuous to the second number of the array and so on. Linear search is not very effective as it is very slow, it goes through all the data individually.

Start program

SIZE = 10

MOD = 4

MAXST = 42

struct all surnames

{

char surnameslist[SIZE]

}

struct names

{

char firstname[SIZE]

char surname[SIZE]

}

struct modules

{

char code[SIZE]

int type

int maximum

int current

struct names students[MAXST]

}

start main

struct modules mods[MOD]= {{"DT265A", 0, 13, 0, {' ', ' '}},

{"DT265C", 0, 9, 0, {' ', ' '}},

{"DT265B", 1, 14, 0, {' ', ' '}},

{"DT8900", 1, 6, 0, {' ', ' '}}}

struct allsurnames surnames[MAXST];

int choice;

int end = 1;

char searchkey[SIZE];

DO

print {Menu 1.join 2.leave 3.Sorted Surnames 4.Display data 5.FULL-TIME students 6.what is your module 7.exit }

input choice

switch (choice)

case 1

join(mods)

break

case 2

leave(mods)

break

case 3

sorted\_surnames(surnames,mods)

break

case 4

display(mods)

break

case 5

PRINT "FULL-TIME students"

linear\_search(mods)

break

case 6

PRINT "enter you surname"

INPUT searchkey

int find = -1

int result

FOR i = 0 , i < MOD , i = i +1

IF (mods[i].current > 0 )

result = binary\_search(mods[i].students,mods[i].current, searchkey)

IF (result NOT -1)

find = i

break

END IF

END IF

END FOR

IF (find NOT -1)

PRINT "Name has been found"

END IF

ELSE

PRINT "studentnot found"

END ELSE

break

case 7

end = -1

break

default

PRINT "choose number form the menu"

break

END SWITCH

END DO WHILE (end == 1 )

END MAIN

START join (struct modules POINTER modsf)

int modch = 0

char mchoice[SIZE]

int find = 0

char firstname[SIZE]

char surname[SIZE]

INPUT mchoice, firstname, surname

FOR i= 0, i < MOD ,i= i+1

modch = compare(modsf[i].code , mchoice)

IF (modch == 0)

find = 1

IF (modsf[i].current < modsf[i].maximum)

modsf[i].current = modsf[i].current + 1

COPY (firstname INTO modsf[i].students[modsf[i].current - 1].firstname)

COPY (surname INTO modsf[i].students[modsf[i].current - 1].surname)

PRINT "you have been added to the module"

END IF

ELSE

PRINT "module is full"

END ELSE

break

END IF

END FOR

IF (find == 0 )

PRINT "choose an exsisting module"

END IF

END FUNCTION JOIN

START leave(struct modules POINTER modsf)

int modch = 0

int namexist = 0

char mchoice[SIZE]

int find = 0

char surname[SIZE]

INPUT mchoice,surname

FOR (i= 0 , i < MOD, i = i +1)

modch = COMPARE (mchoice TO modsf[i].code)

IF (modch = 0)

find = 1

FOR j=0, j <modsf[i].current, j = j+1

nameexist = COMPARE(surname TO modsf[i].students[j].surname)

IF (nameexist = 0 )

modsf[i].current = modsf[i].current - 1

FOR k = 0 , k < modsf[i].current, k = k +1

COPY (modsf[i].students[k + 1].surname INTO modsf[i].students[k].surname )

END FOR

PRINT "you have exited the module"

BREAK

END IF

END FOR

END IF

END FOR

IF find = 0

PRINT "please chose an existing module"

END IF

END FUNCTION leave

START display (struct modules POINTER mods)

FOR i = 0, i < MOD , i = i +1

PRINT "mods[i].code,mods[i].type,mods[i].maximum,mods[i].current"

FOR j= 0 , j < mods[i].current, j = j +1

PRINT mods[i].students[j].firstname,mods[i].students[j].surname

END FOR

END FOR

END DISPLAY FUNCTION

START linear\_search(struct modules POINTER modsf)

int key = 1

found = 0

FOR i =0, i < Mod , i = i +1

FOR j=0, j <MAXST, j = j +1

IF (modsf[i].type == key)

PRINT "modsf[i].students[j].firstname,modsf[i].students[j].surname"

found = 1

BREAK

END IF

END FOR

IF found = 0

PRINT "NO students in full time modules"

END IF

END FOR

END FUNCTION