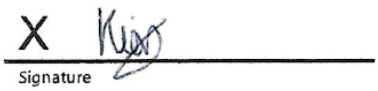
DSA-assignment

Kian Parnis 0107601L

14th April 2021

Statement of Completion

- Question 1 - Attempted and works well.
- Question 2 - Attempted and works well.
- Question 3 - Attempted and works well.
- Question 4 - Attempted and works well.
- Question 5 – Attempted and works well.
- Question 6 - Attempted but does not work fully. (Will be explained during testing portion)
- Question 7 - Attempted and works well.
- Question 8 - Attempted and works well.
- Question 9 - Attempted and works well.
- Question 10 - Attempted and works well.
- Question 11 - Attempted and works well.
- Question 12 - Attempted and works well.



Programming language used.

- C
  - Using Clion as the ide.

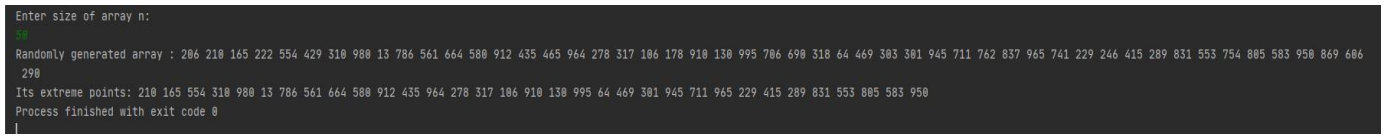
Sample screen dumps and testing.

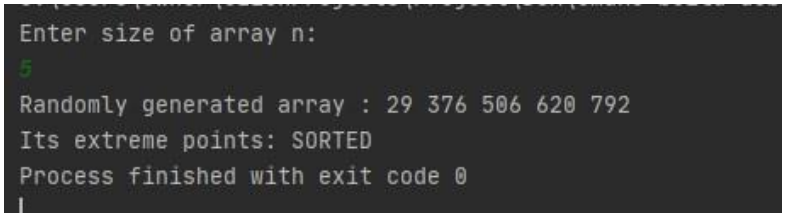
Question 1 and 2

Shell, Quick and Merge sort where all tested by running the code several times and checking if the arrays where all sorted correctly.

Question 3

For extreme points several array lengths were used to check whether the conditions held, if the array given was without extreme points i.e., only if the array were sorted the word 'SORTED' would be printed and if extreme points where present then extreme points would be printed.





Question 4

Varying list sizes were used to randomly generate and check if a 2-pair was found or not, at which the calculations.
would be checked independently (example 393 * 800 = 314400 and 524*600 = 314400) to see whether the answer would hold.

```
Enter the length of list
30
Randomly generated list:
483 474 411 698 356 393 471 137 928 474 156 82 600 454 544 95 961 800 601 23 928 262 524 197 341 893 277 149 425 13

1. (393,800) , (524,600) which is equal to 314400

Process finished with exit code 0
```

```
Enter the length of list
30
Randomly generated list:
650 794 118 166 66 979 481 364 258 280 829 283 194 533 842 625 607 445 779 696 455 289 340 562 636 642 553 749 133 282

There are no 2-pairs in this list!
Process finished with exit code 0
```

Question 5

The program was given several expressions and the conversion to RPN would work and give a correct evaluation as follows:

```
Enter an arithmetic expression in infix:
7*(9+2)
RPN format: 792+*
77.00
Process finished with exit code 0
```

```
Enter an arithmetic expression in infix:
3/(9-2)
RPN format: 392-/
0.43
Process finished with exit code 0
```

```
Enter an arithmetic expression in infix:
9*3
RPN format: 93*
27.00
Process finished with exit code 0
```

```
Enter an arithmetic expression in infix:
+-*
Syntax error
Enter an arithmetic expression in infix:
&^$#$
Error input invalid: &
```

However, the program would occasionally not work as intended and break giving a failed outcome:

```
Enter an arithmetic expression in infix:
9*(7-4)+(2-3)
RPN format: 974-23-+*□□
18.00
Process finished with exit code 0
```

```
Enter an arithmetic expression in infix:
2+7-8
RPN format: 278+
2.00
Process finished with exit code 0
```

Attempts were made to tried to fix the following errors, but no solution was able to be found.

Question 6

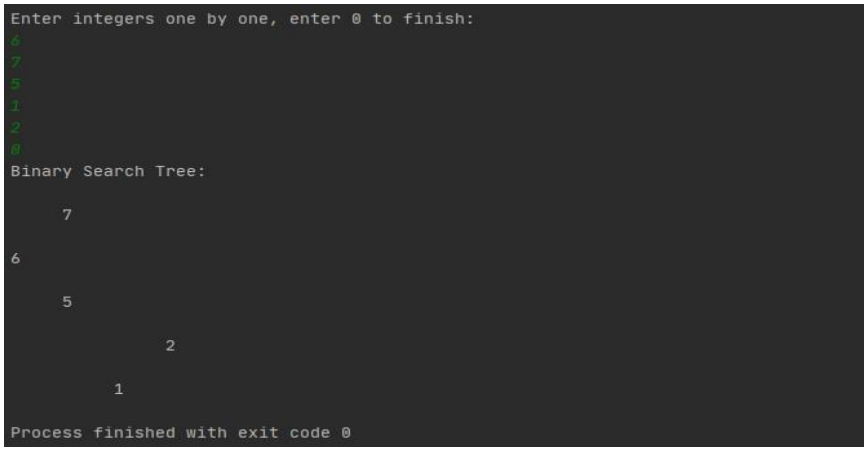
Random numbers were used to check whether the Boolean function correctly identified the prime number and if the validity of the algorithm would hold.

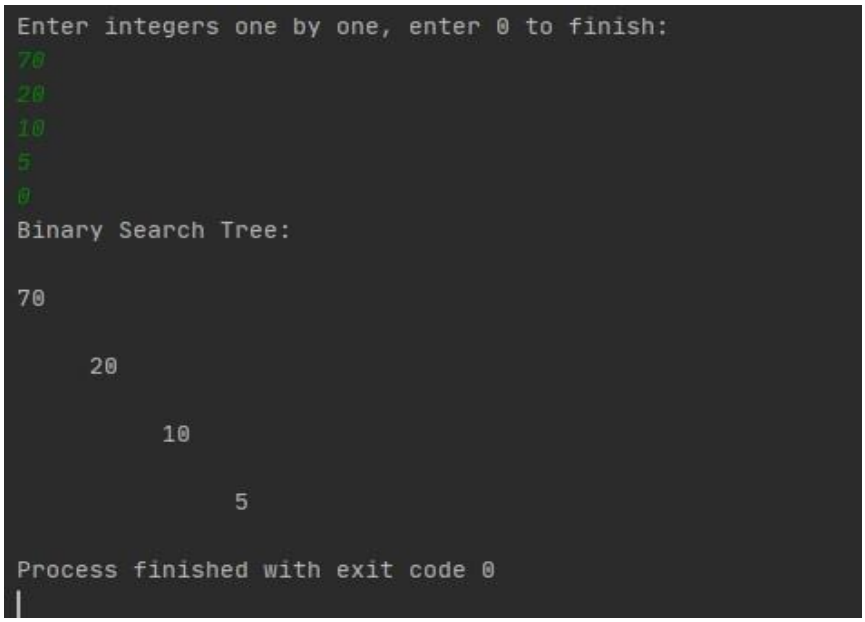
```
Enter a number to check whether or not its prime
89
Is a prime number
Sieve of Eratosthenes: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89
Process finished with exit code 0
```

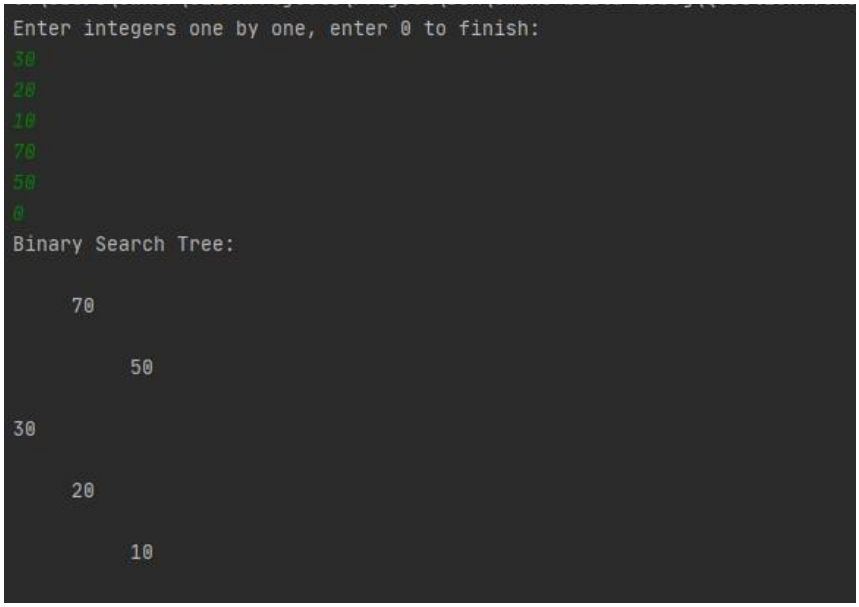
```
Enter a number to check whether or not its prime
77
Isnt a prime number
Sieve of Eratosthenes: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
Process finished with exit code 0
```

Question 7

Random sequence of integers was inputted one by one to check if the binary tree would print successfully. The BTS would print the values with the root being on the RHS and building towards the left with the placements following suit.

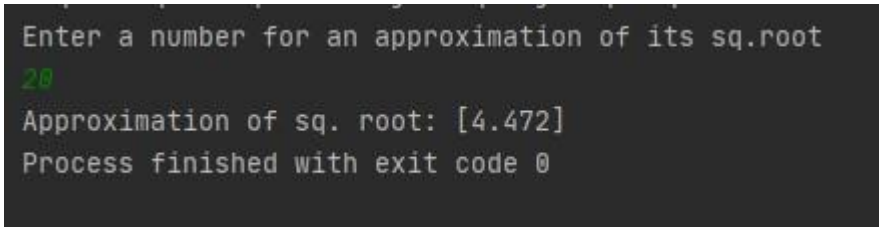


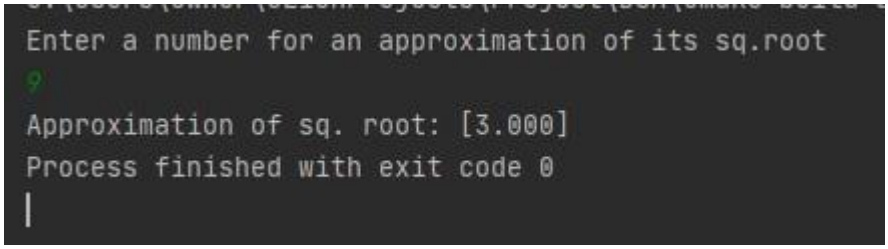


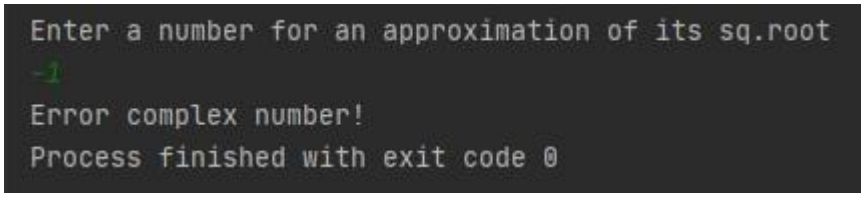


Question 8

A set of whole numbers were tested to check whether the program would work successfully, these were also tested independently to check if the approximation were correct. For the set of negative numbers, an error would be brought up stating that complex numbers cannot be calculated.







Question 9

The size of the array was reduced during testing and each value was independently checked and counted to see whether the programs result held correctly.

```
707 878 986 1003 303 416 572 728 468 410 394 784 479 485 321 702 388 35 981 897 173 206 1022 974 178 213 312 89 301 791 310 197 679 93 250 727 18 26 764 322 452 566 303 654 734 613 607 351 470 244 438 578 427 76 187 684 735
 945 671 141 444 54 841 134 825 399 700 953 55 334 252 138 603 698 474 759 79 31 275 864 826 294 820 563 633 679 458 195 669 35 155 840 454 321 702 145 568 475 72 515 925 727 897 641 606 887 889 150 321 419 463 903 381 373
 339 147 778 916 159 491 157 14 263 754 610 719 488 420 298 125 68 195 142 516 496 195 108 98 528 951 241 334 266 280 912 766 244 673 966 636 664 484 373 435 467 435 211 328 836 135 469 54 459 869 265 121 957 345 811 315
 380 187 588 478 769 419 478 705 782 432 198 765 279 93 467 760 199 493 527 394 273 526 430 103 908 329 891 1004 91 437 342 623 400 996 171 449 89 288 980 515 106 788 535 132 392 333 584 515 0 508 855 965 473 792 681 982
 176 401 442 419 781 1013 369 944 598 329 826 903 318 468 267 362 528 24 2 694 348 55 962 194 309 1015 477 155 11 79 543 426 22 886 564 829 853 448 568 347 912 352 242 437 548 701 320 608 183 485 642 293 524 298 654 292 627
 593 674 163 477 647 523 207 118 622 766 550 158 202 616 817 339 122 314 716 932 647 43 402 404 446 516 827 215 959 924 377 411 823 790 102 492 641 966 115 220 424 546 32 607 723 126 339 850 705 780 347 478 998 324 79 334
 287 617 113 838 985 885 886 137 241 23 94 637 263 293
Value: [35], Amount Repeated: [1]
Value: [54], Amount Repeated: [1]
Value: [55], Amount Repeated: [1]
Value: [79], Amount Repeated: [2]
Value: [89], Amount Repeated: [1]
Value: [93], Amount Repeated: [1]
Value: [155], Amount Repeated: [1]
Value: [187], Amount Repeated: [1]
Value: [195], Amount Repeated: [2]
Value: [241], Amount Repeated: [1]
Value: [244], Amount Repeated: [1]
Value: [263], Amount Repeated: [1]
Value: [293], Amount Repeated: [1]
Value: [301], Amount Repeated: [1]
Value: [303], Amount Repeated: [1]
Value: [321], Amount Repeated: [2]
Value: [329], Amount Repeated: [1]
Value: [334], Amount Repeated: [2]
Value: [339], Amount Repeated: [2]
Value: [347], Amount Repeated: [1]
Value: [373], Amount Repeated: [1]
Value: [394], Amount Repeated: [1]
Value: [419], Amount Repeated: [2]
```

Question 10

Varying sizes of array lengths were used to check whether the recursive function could successfully the largest number, given the base condition was one, if a single length were inputted that same value would be returned right away while if the length is less then one an error would be given.

```
Enter the length of list
30
Randomly generated list:
841 235 253 459 292 647 185 507 590 233 480 540 887 1007 864 918 164 780 255 818 264 65 188 835 876 1021 502 789 397 1011
Largest num is: 1021

Process finished with exit code 0
```

```
Enter the length of list
0
Please enter a number greater then 0
1
Randomly generated list:
748
Largest num is: 748
```

Question 11

Both the values of degree's and n'terms were varied to check whether the function held true, the library 'math.h ' was also used to check the accuracy of the results.

```
Enter value of degree's:
90
Enter value n:
4
By Math.h -> cos(90.00) = 0.0000
By Maclaurin -> cos(90.00) = 0.0200
By Math.h -> sin(90.00) = 1.0000
By Maclaurin -> sin(90.00) = 0.9248

Process finished with exit code 0
```

```
Enter value of degree's:
30
Enter value n:
20
By Math.h -> cos(30.00) = 0.8660
By Maclaurin -> cos(30.00) = 0.8660
By Math.h -> sin(30.00) = 0.5000
By Maclaurin -> sin(30.00) = 0.5000

Process finished with exit code 0
```

Question 12

Different lengths of n numbers were used to check both iteratively and recursively the first n numbers of the Fibonacci sequence and the sum.

```
Please enter a value that's greater or equal to 1
9
Recursive Fibonacci of 9: 1 1 2 3 5 8 13 21 34
Iterative Fibonacci of 9: 1 1 2 3 5 8 13 21 34
Sum of the first 9 nums: 88
Process finished with exit code 0
```

```
Please enter a value that's greater or equal to 1
0
Value doesn't satisfy conditions
Please enter a value that's greater or equal to 1
30
Recursive Fibonacci of 30: 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040
Iterative Fibonacci of 30: 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040
Sum of the first 30 nums: 2178308
Process finished with exit code 0
```

Question 1 and 2

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

#define SHELL_ARR 301
#define QUICK_ARR 353
#define MERG_ARR (SHELL_ARR + QUICK_ARR)
#define RAND 1025

void s_sort(int array[]); void  q_sort(int
array[]);

void shellsort(int array[], int k); void quicksort(int array[], int f, int l); void  mergesort(const
int arr_A[],const int arr_B[], int arr_C[]);

int part(int array[], int f, int l); void swap(int* x, int* y);  int
main(void)
{ setvbuf(stdout, NULL, _IONBF, 0); srand(time(NULL));

        static int A[SHELL_ARR]; static int  B[QUICK_ARR];
        static int C[MERG_ARR];

        s_sort(A); q_sort(B);
        mergesort(A,B,C); printf("ShellSort:
        ");

        for(int i=0;i<SHELL_ARR;i++) { printf(("%d
             "),          A[i]);      }
        printf("\nQuickSort: ");

        for(int i=0;i<QUICK_ARR;i++) {
        printf(("%d      "),      B[i]);      }
        printf("\nMergeSort: ");

        for(int i=0;i<MERG_ARR;i++) { printf(("%d
               "), C[i]);
        }
        return 0; }

void s_sort(int array[])
{
        /* populating with randomly generated integers */ for(int i=0;i<SHELL_ARR;i++)
        {

                array[i] = rand() % RAND; }
        int g = SHELL_ARR/2;
        shellsort(array, g);
}
```

```
void shellsort(int array[], int k){ while (k > 0){ int j; for(int i = k;
        i < SHELL_ARR; i++){ int temp = array[i];  for(j = i; j >= k &&
        array[j - k] > temp; j -= k){ array[j] = array[j - k];
                } array[j] = temp;
                }

                if (k == 2){ k =
                        1;
                }

                else{ k = k/2;
                } } }


void q_sort(int arr[])
{
      /* populating with randomly generated integers */ for(int
        i=0;i<QUICK_ARR;i++) { arr[i] = rand() % RAND;
        }

        int first = 0; int last = QUICK_ARR -
      1; quicksort(arr, first , last);

} void quicksort(int arr[], int f, int l){

  if (f < l)
  { int Position = part(arr, f, l);

          quicksort(arr, f, Position-1); quicksort(arr, Position+1,
          l);
  }
} int part(int array[], int f, int l){

        int pivot = array[l]; int j = (f -
        1);

        for(int i = f; i <= l - 1; i++){ if(array[i] < pivot){ j++; swap(&array[j],
                &array[i]);
                }
        }
        swap(&array[j+1], &array[l]);

        return (j+1);
}

void swap(int* x, int* y)
{
        int temp = *x; *x = *y;
        *y = temp;
```

```
} void mergesort(const int arr_A[], const int arr_B[], int arr_C[]){ int point_A = 0, point_B = 0, point_C =

0;

        do{ if (arr_A[point_A] <= arr_B[point_B]) { arr_C[point_C++]
                = arr_A[point_A++];
                } else {
                        arr_C[point_C++] = arr_B[point_B++];
                }

        }while(point_A < SHELL_ARR && point_B < QUICK_ARR);

        while(point_A < SHELL_ARR){ arr_C[point_C++] = arr_A[point_A++];
        }

        while (point_B < QUICK_ARR){ arr_C[point_C++] = arr_B[point_B++];
        }

}
```

Question 3

```
#include <stdlib.h> #define RAND 1025 void extreme_pts(int array[],

int length);

int main(void)
{ setvbuf(stdout, NULL, _IONBF, 0); srand(time(NULL));
        int n = 0; int
        count = 0;

        printf("Enter size of array n: \n"); scanf("%d",
        &n); int A[n];

        extreme_pts(A, n); printf("Its extreme
        points: ");

        for(int i=1;i<n-1;i++) { if((A[i] > A[i+1] && A[i-1] < A[i]) || (A[i] < A[i+1] && A[i-1] > A[i])) { printf(("%d "), A[i]);
                count++;
                }

        } if(count==0) { printf("SORTED");
        } return 0;
}

void extreme_pts(int array[], int length) {
        /* populating with randomly generated integers */ for(int i=0;i<length;i++)
        {

                array[i] = rand() % RAND; }
        printf("Randomly generated array : ");
```

```
    for(int i=0;i<length;i++) { printf(("%d "), array[i]); } printf("\n");


}
```

Question 4

```
#include <stdlib.h> #define RAND 1023 void ran_gen(int array[],

int length);

int main(void)
{ setvbuf(stdout, NULL, _IONBF, 0); srand(time(NULL)); int len = 0; printf("Enter
    the length of list\n"); scanf("%d", &len);  while(len<=0){ printf("Please enter
    a number greater then 0\n"); scanf("%d", &len);
    } int B[len]; ran_gen(B, len);

    return 0;
}
void ran_gen(int arr[], int length) {
    /* populating with randomly generated integers */ for(int
    i=0;i<length;i++)       {           arr[i]      = (rand()           %
        RAND)+1;           } printf("Randomly generated list: \n");

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

    for(int i=0;i<length;i++) { for(int
        j=0;j<length;j++){ if(i==j || i>j){
        matrix[i][j] = 0;
            } else { matrix[i][j] = arr[i] * arr[j]; }
          // printf("%d ",matrix[i][j]);
        }
        //printf("\n"); }

    int store; int
    count=0;
    for(int i=0;i<length;i++) { for(int j=0;j<length;j++){ if
        (i<j) { store = matrix[i][j];

                for(int k=(i+1);k<length;k++) { for(int l=0;l<length;l++) { if(store==matrix[k][l] && store!=0 && arr[i] != arr[l]
                    && arr[i] != arr[k]){ count++; printf("%d. (%d,%d) , (%d,%d) which
                        is equal to %d \n",count,
                        arr[i],arr[j],arr[l],arr[k], matrix[k][l]); matrix[k][l] = 0; matrix[i][j] = 0;
                            }
                    }
                }

            }
```

```
        } } if(count==0){ printf("There are no 2-pairs in this list!");
    }


}
```

Question 5

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> #define MAX

100

int StackFull(const int *StackTop); int StackEmpty(const int *StackTop); char
StackPush(char var, int *StackTop, char array[]); double StackPushCal(double var,
int *StackTop, double array[]); double StackPopCal(int *StackTop, const double
array[]); char StackPop(int *StackTop, const char array[]); void infixtoRPN(char
array[], char Rev[]); void calc(char Rev[]);

int main(void)
{ setvbuf(stdout, NULL, _IONBF, 0); char str[MAX]; char RPN[100];
    int num_present = 0; label: printf("Enter an arithmetic
    expression in infix:\n"); scanf("%s",
    &str); for(int i=0; i<strlen(str); i++){

        if(str[i]>47 && str[i]<58){ num_present = 1;
        }
        else if ((str[i]>39 && str[i]<44) || str[i]==45 || str[i] ==47){

        }
        else{ printf("Error input invalid: "); printf("%c\n", str[i]);
            goto label;
        }

        if(num_present == 0){ printf("Syntax error\n"); goto
            label;
        } } infixtoRPN(str, RPN); printf("RPN format:

    ");

    for(int i=0; i<strlen(RPN); i++){ printf("%c",
    RPN[i]); } printf("\n"); calc(RPN); return 0; }

void calc(char Rev[]){ int Top = 1;
    double first, last; double
    stack_2[100];
```

```
    for(int i=0; i<strlen(Rev); i++) { if(Rev[i] == 43 || Rev[i] == 45 || Rev[i] == 42 || Rev[i] ==
        47){ first = StackPopCal(&Top, stack_2); last = StackPopCal(&Top, stack_2);
        if(Rev[i]==43){
                        StackPushCal((last+first), &Top, stack_2);
            }
            else if(Rev[i]==45){
                        StackPushCal((last-first), &Top, stack_2);
            }
            else if(Rev[i]==42){
                        StackPushCal((last*first), &Top, stack_2);
            } else{
                        StackPushCal((last/first), &Top, stack_2);
            }
        }else{
            StackPushCal(Rev[i]- 48 , &Top, stack_2);
        }
    } printf("%.2lf",stack_2[0]);

}


void infixtoRPN(char array[], char Rev[]){ char stack[100]; char
    pre, temp; int Top = -1, k = 0;

    for(int i=0; i<strlen(array); i++) { if (array[i] > 47 && array[i] < 58) { //between
        0-9 Rev[k] = array[i]; k++;
        } else if ((array[i] < 48 || array[i] > 57) && StackEmpty(&Top)) { //other than 0-9 and stack is e pre = StackPush(array[i],
        &Top, stack);
        }
        else if (array[i] == 40) { // ( always put pre = StackPush(array[i], &Top,
            stack);
        } else if (array[i] == 41) { //always pop at ) do{ temp=StackPop(&Top, stack);

                if(temp!=40) { Rev[k] = temp; k++;
                }
                pre = 40;
                if(k==100){ printf("Error incorrect brackets"); exit(-1);
                }
            }while(temp!=40);
        }
        else if(pre == 40 && (array[i] == 42 || array[i] == 43 || array[i] == 45 || array[i] == 47)){ // pre = StackPush(array[i], &Top,
            stack);
        }
        else if((pre == 43 || pre == 45) && (array[i] == 42 || array[i] == 47)){ //* and - highest prece pre = StackPush(array[i], &Top
    stack); } }

    while(!StackEmpty(&Top)){
        Rev[k] = StackPop(&Top, stack); k++;
    } }
```

```
int StackFull(const int *StackTop){ if(*StackTop ==
        MAX){
                return 1; }
        else{ return 0;
        } }

int StackEmpty(const int *StackTop){
        if(*StackTop == -1){ return 1; } else{ return
        0;
        } }
char StackPush(char var, int *StackTop, char array[]){
        if(!StackFull(StackTop)){ *StackTop = *StackTop+1;
        array[*StackTop] = var; return var; }else{ return 0;
        } }

double StackPushCal(double var, int *StackTop, double array[]){
        if(!StackFull(StackTop)){ *StackTop = *StackTop+1; array[*StackTop] = var; return
        var;
        }else{ return 0;
        } }

char StackPop(int *StackTop, const char array[]){ char
        temp_var; if(!StackEmpty(StackTop)){ temp_var =
        array[*StackTop]; *StackTop = *StackTop -1;

                return temp_var;
        } else{ return 0;
        } }

double StackPopCal(int *StackTop, const double array[]){ double temp_var; if(!StackEmpty(StackTop)){
        temp_var = array[*StackTop]; *StackTop =
                *StackTop -1;

                return temp_var;
        } else{ return 0;
        }
}
```

Question 6

```
#include <stdio.h>
#include <stdbool.h> #include
<math.h>

void SieveofEratosthenes(int n); bool func(int  isprime);

int main(void)
{ int num; setvbuf(stdout, NULL, _IONBF, 0); printf("Enter a number to check whether or not

        its prime\n"); scanf("%d", &num);
```

```
        if(func(num)){ printf("Is a prime number\n");
        } else{ printf("Isnt a prime number\n");
        } SieveofEratosthenes(num);  return 0;
        } bool func(int
isprime){

        if (isprime<2) return false; else{ for(int i=2; i<=
        sqrt(isprime); i++){ if(isprime%i==0) return
                 false;
                } return true;
        } } void SieveofEratosthenes(int n){ bool is_prime[n+1];


    for (int i=0; i<=n; i++){ is_prime[i] = 1;
    }

    for(int i = 2; i<=n; i++){ if(is_prime[i]==1){
                  for(int j = i*i; j<=n; j += i){ is_prime[j] = 0;
                  } } }

    printf("Sieve of Eratosthenes: "); for(int i = 2;
    i<=n; i++){ if(is_prime[i] == 1){ printf("%d ",i);
           }
    }

}
```

Question 7

```
#include <stdio.h>
#include <malloc.h>
#define AMOUNT 5 //default space amount
struct BST
{ int node; struct BST* left, *right;
};

void display(struct BST *root, int space); struct BST*  addchild(struct
BST *root, int val); struct BST* resize(int val);

int main()
{ setvbuf(stdout, NULL, _IONBF, 0); printf("Enter integers one by one, enter 0 to finish:
        \n"); int value; struct BST *root; int flag=0;

        do{ scanf("%d", &value); if(flag==0 && value!=0)
                { root =
                resize(value); flag=1;
                }
                else if(value!=0) { addchild(root, value);
                }
        }while (value != 0); printf("Binary Search Tree:
```

```
\n"); display(root, 0); return 0; }

void display(struct BST *root, int space)
{ if (root == NULL) { //Base case return;
}

space += AMOUNT; display(root->right,
space); printf("\n");

for (int i = AMOUNT; i < space; i++) { printf("
"); } printf("%d\n",  root->node);

display(root->left, space);
}

struct BST* addchild(struct BST *root, int val)
{ if(root==NULL) { return resize(val);
} else if(val>root->node){ root->right =
addchild(root->right, val);  } else { root->left
= addchild(root->left, val); } return root; }

struct BST* resize(int val)
{ struct BST *node = malloc(sizeof(struct BST)); node->node = val; node->left
= node->right = NULL;

return node;
}
```

Question 8

```
#include  <stdio.h>  #define
LARGE_NUM  9999  double
newt_approx(double num);

int main(void)
{ setvbuf(stdout, NULL, _IONBF, 0); double num
= 0; printf("Enter a number for an approximation of its sq.root\n"); scanf("%lf",

&num);

double value = newt_approx(num); if(value == -1){ printf("Error
complex number!");
} else{ printf("Approximation of sq. root: [%.3lf]", value);
} return 0; } double newt_approx(double num){
if(num ==
0){ return num;

} else if(num <= -1){ return -1; } double difference = LARGE_NUM, guess,

update_guess, err =  0.001; guess = num;
```

```
    do{ update_guess = guess - (((guess*guess) - num) / (guess*2)); difference = update_guess - guess;

        if(difference < 0){ difference *= -1;
        } guess = update_guess; }while(difference

    > err);


    return guess;
}
```

Question 9

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

#define QUICK_ARR 353 #define RAND
1025

void q_sort(int array[]); void quicksort(int array[], int  f,
int l); int part(int array[], int f, int l); void swap(int* x,
int* y); void repeat(const int array[]);
int main(void)
{ setvbuf(stdout, NULL, _IONBF, 0);
    srand(time(NULL)); static int
    B[QUICK_ARR]; q_sort(B); return
    0; }

void q_sort(int arr[])
{
    /* populating with randomly generated integers */
    for(int i=0;i<QUICK_ARR;i++) { arr[i] = rand() % RAND;
    }

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

    int first = 0; int last =  QUICK_ARR - 1;
    quicksort(arr, first , last); repeat(arr);

} void quicksort(int arr[], int f, int l){

    if (f < l)
    {
        int Position = part(arr, f, l);

        quicksort(arr, f, Position-1); quicksort(arr, Position+1,
```

```
            l);
        }

} int part(int array[], int f, int l){

        int pivot = array[l]; int j = (f -
        1);

        for(int i = f; i <= l - 1; i++){ if(array[i] < pivot){ j++; swap(&array[j],
            &array[i]);
            } } swap(&array[j+1], &array[l]);

        return (j+1);
}

void swap(int* x, int* y)
{
        int temp = *x; *x = *y;
        *y = temp;
}

void repeat(const int array[]) { int a = 0, amount = 0, num_rep =  0, old_a = 0; do {

            if(amount != 0 && old_a == amount){ printf("Value: [%d], Amount Repeated: [%d]\n", array[a-1], amount);
                num_rep++; amount = 0;
            } old_a = amount;

            if(array[a] == array[a+1]) { amount++;
            }

            a++;
        } while (a != QUICK_ARR); if(num_rep==0){ printf("Array doesn't contain any
        repetitions!"); }
}
```

Question 10

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h> #define RAND

1025

void ran_gen(int array[], int length); int  large_num(int
array[], int length);

int main(void)
{ setvbuf(stdout, NULL, _IONBF, 0); srand(time(NULL));
        int len = 0; printf("Enter the length of list\n");
```

```
    scanf("%d", &len); while(len<=0){ printf("Please enter a number
    greater then 0\n"); scanf("%d", &len);  } int B[len]; ran_gen(B, len);
    return 0; }

void ran_gen(int arr[], int length)
{
    /* populating with randomly generated integers */ for(int i=0;i<length;i++)
        {       arr[i]   =   rand()   %   RAND;   }
    printf("Randomly generated list: \n");

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

    printf("Largest num is: %d\n", large_num(arr, length));
}

int large_num(int array[], int length){ int prev, curr;

    if(length==1){ return array[length1];
    }
    else{ prev = large_num(array, length-1); curr =
        array[length-1]; if (prev > curr){ return prev;
        } else{ return curr;
        }
    }

}
```

Question 11

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

double sin_cal(int n, double val); double
cos_cal(int n, double val); double power(double
val, int n); long factorial(int n);

int main(void)
{ setvbuf(stdout, NULL, _IONBF, 0); double x, radian; double
    result;

    x = 0; printf("Enter value of degree's:\n");
    scanf("%lf",  &x);

    radian = x*(3.14159/180.0); int pow =
    0; printf("Enter value n:\n"); scanf("%d",

    &pow);

    result = cos(radian); printf("By Math.h -> cos(%.2lf) = %.4lf\n", x,
    result);
```

```
    if(pow % 2 != 0){ pow--;
    }

    result = cos_cal(pow,radian); printf("By Maclaurin -> cos(%.2lf) = %.4lf\n",
    x, result);

    result = sin(radian); printf("By Math.h -> sin(%.2lf) = %.4lf\n", x,
    result); if(pow % 2 == 0){ pow--;
    }

    result = sin_cal(pow,radian); printf("By Maclaurin -> sin(%.2lf) = %.4lf\n",
    x, result); return

    0; }

double sin_cal(int n, double val){ if(n==1){ return val;
    }
    if((n-1)/2 % 2 == 0){ return (sin_cal(n-2,val) + (power(val,n)/factorial(n))); } else{
    return (sin_cal(n-2,val) - (power(val,n)/factorial(n))); } }

double cos_cal(int n, double val){ if(n==0){ return
    1;
    }

    if((n-1)/2 % 2 != 0){ return (cos_cal(n-2,val) + (power(val,n)/factorial(n)));
    } else{ return (cos_cal(n-2,val) - (power(val,n)/factorial(n))); } } double power(double val,
int n){

    if(n==0){ return 1; }
    else{ double num = 1;

        for(int i=0; i<n; i++){ num *= val;
        } return num;
    }

}
long factorial(int n)
{
    if (n == 0) { return 1; } else { return (n *
    factorial(n - 1));
    }
}
```

Question 12

```
#include <stdio.h> unsigned long long
fibonacciIt(int n); unsigned long long
fibonacciRe(int n); int main(void) {
setvbuf(stdout, NULL, _IONBF,0);
```

```
    int num = 0; unsigned long long temp , str = 0; label: printf("Please enter a value
    that's greater or equal to 1\n"); scanf("%d", &num); if(num >= 1) {
    printf("Recursive Fibonacci of %d: ", num);

        for(int i = 0;i<num; i++) { printf("%d ", fibonacciRe((i+1)));
        } printf("\n");

        printf("Iterative Fibonacci of %d: ", num); for(int i =
        0;i<num; i++) { temp = fibonacciIt((i+1)); str += temp;
        printf("%d ", temp); } printf("\n");  printf("Sum of
        the first %d nums: %d", num, str);
    } else{ printf("Value doesn't satisfy conditions \n"); goto label;
    }


    return 0; }
unsigned long long fibonacciIt(int n)
{
    unsigned long long prepre, pre = 0, nm =1; for(int i

  =1; i<n ;i++)

  { prepre = pre; pre = nm; nm
        =       prepre+pre;
        }
  return nm; }

unsigned long long fibonacciRe(int n)
{ if(n <= 1) { return n;
    }
    else {
                        return (fibonacciRe(n-1) + fibonacciRe(n-2));
    }

}
```

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Declaration

Plagiarism is defined as "the unacknowledged use, as one's own, of work of another person, whether or not such work has been published, and as may be further elaborated in Faculty or University guidelines" (University Assessment Regulations, 2009, Regulation 39 (b)(i), University of Malta).

I / We*, the undersigned, declare that the [assignment / Assigned Practical Task report / Final Year Project report] submitted is my / our* work, except where acknowledged and referenced.

I / We* understand that the penalties for committing a breach of the regulations include loss of marks; cancellation of examination results; enforced suspension of studies; or expulsion from the degree programme.

Work submitted without this signed declaration will not be corrected, and will be given zero marks.

* Delete as appropriate.

(N. B. If the assignment is meant to be submitted anonymously, please sign this form and submit it to the Departmental Officer separately from the assignment).

Kian Parnis	Kian
Student Name	Signature
Student Name	Signature
Student Name	Signature
Student Name	Signature

ICT 1018	Data Structures and Algorithms I
Course Code	Title of work submitted

5/31/2021

Date