MATH 152 Lab 7

Put team members' names and section number here.

```
import sympy as sp
from sympy.plotting import (plot,plot_parametric)
import matplotlib.pyplot as plt

n, k = sp.symbols('n k', integer=True, positive=True)
i = sp.I
```

Question 1

1a

```
# Enter your code here
sk_r = sp.summation(sp.sin(n), (n, 1, k))
sk_sim = sp.simplify(sk_r)

print("s_k (raw):", sk_r)
print("s_k (simplified):", sk_sim)
```

```
s_k (raw): Sum(sin(n), (n, 1, k))
s_k (simplified): Sum(sin(n), (n, 1, k))
```

1b

```
# Enter your code here
# (b)
tk = sp.summation(sp.exp( i*n ), ( n, 1, k ))
tk = sp.simplify(tk)
print("t_k (closed form):", tk)
```

```
t_k (closed form): (-exp(I*(k + 1)) + exp(I))/(1 - exp(I))
```

1c

```
# Enter your code here
# (c)
sk_from_tk = sp.simplify( sp.im(tk).rewrite(sp.sin) )
print("Im(t_k) =", sk_from_tk)
```

```
Im(t_k) = (sin(k) - sin(k + 1) + sin(1))/(2 - 2*cos(1))
```

1d

```
# Enter your answer here

sk_closed = sp.sin(k/2) * sp.sin((k+1)/2) / sp.sin(sp.Rational(1, 2))

upp_bd = 1 / sp.Abs(sp.sin(sp.Rational(1, 2)))

print("|s_k| ≤", upp_bd.evalf())

# Since (1/n) is decreasing to 0 and the partial sums s_k are bounded, Dirichlet's test
# implies that sum_{n=1}^\infty sin(n)/n converges.
```

```
|s_k| ≤ 2.08582964293349
```

Question 2

2a

```
# Enter your answer here
from math import log

n, k = sp.symbols('n k', integer=True, positive=True)
i = sp.I


sN = sp.summation((-1)**n, (n, 0, k))
print("s_k =", sp.simplify(sN))


gr1 = sp.summation(((-1)**(2*n)) + ((-1)**(2*n+1)), (n, 0, k))
print("Grouped as (1-1)+...  =", sp.simplify(gr1))

gr2 = 1 + sp.summation(((-1)**(2*n+1)) + ((-1)**(2*n+2)), (n, 0, k))
print("Grouped as 1+(-1+1)+... =", sp.simplify(gr2))
```

```
s_k = (-1)**k/2 + 1/2
Grouped as (1-1)+...  = 0
Grouped as 1+(-1+1)+... = 1
```

2b

```
# Enter your answer here
a = sp.Function('a')
a_n = sp.Piecewise((1/n, sp.Eq(n % 2, 1)), (-1/n**2, True))

def partial_sum_A(N):
    return sum((1/j if j % 2 == 1 else -1/j**2) for j in range(1, N+1))

A2_symbolic = sp.summation(a_n.subs(n, 2*n), (n, 1, sp.oo))
print("Sum of even terms exactly:", sp.simplify(A2_symbolic))

print("Partial sums of Σ a_n (first few N):")
for N in [50, 200, 1000]:
    print(N, float(partial_sum_A(N)))
```

```
Sum of even terms exactly: -pi**2/24
Partial sums of Σ a_n (first few N):
50 2.1897953985549212
200 2.875596214255411
1000 3.678325129176359
```

2c

```
# Enter your code here
alt_harmonic = sp.summation((-1)**(n+1)/n, (n, 1, sp.oo))
print("Exact sum:", alt_harmonic)
print("Numeric :", sp.N(alt_harmonic))
```

```
Exact sum: log(2)
Numeric : 0.693147180559945
```

2d

```
# Enter your code here
gk = 1/(4*k - 3) + 1/(4*k - 1) - 1/(2*k)
gk_simp = sp.simplify(sp.together(gk))
print("g k simplified:", gk simp)
```

```
pos = sp.simplify(gk - (8*k - 3)/(2*k*(16*k**2 - 16*k + 3)))
print("Matches (8k-3)/(2k(16k^2-16k+3))= ->", sp.simplify(pos) == 0)
print("Is g_k > 0 for k=1..10? ->", all(gk.subs(k, K).evalf() > 0 for K in range(1, 11)))
```

```
g_k simplified: (8*k - 3)/(2*k*(16*k**2 - 16*k + 3))
Matches (8k-3)/(2k(16k^2-16k+3))= -> True
Is g_k > 0 for k=1..10? -> True
```

2e

```
# Enter your code here
M = 100
S_M = sp.N(sp.summation(gk, (k, 1, M)))
print(f"S_{M} (sum of first {M} groups) ≈", S_M)

targ = sp.Rational(3, 2) * sp.log(2)
print("Target (3/2 * log 2) ≈", sp.N(targ))
print("Under-approximation=", S_M < sp.N(targ))
```

```
S_100 (sum of first 100 groups) ≈ 1.03722545829597
Target (3/2 * log 2) ≈ 1.03972077083992
Under-approximation= True
```

2f

```
# Enter your answer here
S_original = sp.log(2)

S_rearranged = sp.Rational(3, 2) * sp.log(2)

print("Original S =", sp.N(S_original))
print("Rearranged S' =", sp.N(S_rearranged))
print("Equal:", sp.simplify(S_rearranged - S_original) == 0)
print("Difference S'-S =", sp.simplify(S_rearranged - S_original))
```

```
Original S = 0.693147180559945
Rearranged S' = 1.03972077083992
Equal: False
Difference S'-S = log(2)/2
```