UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO´

Corso di Laurea in Informatica Applicata

Programma per la verifica delle congetture di Beal, Cramer e Collatz

Relazione del progetto d'esame per il corso di Programmazione Procedurale
Sessione Invernale 2022/23

Presentata da:

Balducci Milena
Matricola 321791
1°**anno**

0.1 Specifica del Problema

La congettura di Beal asserisce che se $a^x + b^y = c^z$ dove $a,b,c,x,y,z \in \mathbb{N}$ con $a,b,c \geq 1$ e $x,y,z \geq 3$, allora a,b,c hanno un fattore primo in comune.
La congettura di Collatz asserisce che la funzione $f : \mathbb{N}_{>0} \longrightarrow \mathbb{N}_{>0}$ definita ponendo $f(n) = n/2$ se n è pari e $f(n) = 3 \cdot n + 1$ se n è dispari genera 1 dopo un numero finito di applicazioni ai numeri mano a mano ottenuti.
La congettura di Cramer asserisce che il valore assoluto della differenza tra due numeri primi consecutivi è minore del quadrato del logaritmo naturale del più piccolo dei due numeri.
Scrivere un programma ANSI C che chieda all'utente quale congettura intende considerare e poi la verifica acquisendo dalla tastiera a,b,c,x,y,z nel primo caso (se non vale $a^x + b^y = c^z$, il programma lo stampa sullo schermo e poi verifica comunque se a,b,c hanno un fattore primo in comune e ne stampa l'esito sullo schermo), $n > 0$ nel secondo caso (il programma stampa sullo schermo tutti i numeri generati), due numeri primi consecutivi nel terzo caso (il programma stampa sullo schermo sia il valore assoluto della differenza tra i due numeri che il quadrato del logaritmo naturale del più piccolo dei due numeri).

0.2 Analisi del Problema

0.2.1 Dati di Ingresso del Problema

I dati di ingresso sono l'operazione scelta dall'utente e, a seconda della scelta effettuata:

- tre potenze con base intera $b \geq 1$ e esponente intero $e \geq 3$;
- un numero intero n> 0;
- due numeri primi p_n e p_{n+1} consecutivi.

0.2.2 Dati di uscita del Problema

I dati di uscita del problema sono l'esito della verifica della congettura in esame e, in base all'operazione scelta dall'utente:

- l'insieme dei fattori primi comuni alle tre basi scelte;
- l'insieme dei valori generati dalle applicazioni successive della funzione $f : \mathbb{N}_{>0} \longrightarrow \mathbb{N}_{>0}$;
- il valore assoluto della differenza tra i due numeri primi inseriti, e il quadrato del logaritmo naturale del minore dei due.

0.2.3 Relazioni intercorrenti tra i dati del Problema

- I fattori primi di un numero $n \in \mathbb{N}_{>3}$ sono tutti i numeri primi che lo dividono esattamente (ovvero senza resto).
  Per individuare i fattori primi di un numero n è sufficiente effettuare divisioni successive di n per un divisore p tale che $1 < p \leq \sqrt{n}$:
  - Se $n/p \in \mathbb{N}$, si ripete la divisione sul quoziente n/p;
  - Se $n/p \notin \mathbb{N}$, si aumenta p di una unità.

  E' possibile partire da $p = 2$ o $p = \sqrt{n}$, l'insieme di fattori ottenuto non cambia.

- Lo stesso metodo si può utilizzare per individuare i numeri primi: se nessuno dei numeri interi compresi tra 2 e $\sqrt{p}$ è un divisore di p, allora quest'ultimo è un numero primo.
  Due numeri primi p_n e p_{n+1} sono consecutivi se nessuno dei numeri $p_n < n < p_{n+1}$ è primo.

0.3 Progettazione dell'Algoritmo

0.3.1 Scelte di progetto

Per semplificare la validazione dei dati di ingresso nella congettura di Cramer, che richiede di verificare che i due numeri primi inseriti siano consecutivi, si è deciso di imporre dei limiti all'utente, acquisendo prima il numero primo minore e poi quello maggiore. In questo modo è anche possibile diminuire il numero di passi necessario all'algoritmo, in quanto non è necessario verificare quale dei due numeri inseriti sia il minore.

Per l'allocazione dei risultati delle congetture di Beal e Collatz, che producono un numero variabile di valori, si è deciso di creare un tipo di dato strutturato ricorsivo con il quale creare liste di valori di tipo unsigned int, la cui dimensione varia in base al numero di risultati prodotti. Si rende necessario dunque creare due funzioni, una per inserire valori nella lista e una per stamparne il contenuto.

Per evitare ridondanze di codice, si è deciso inoltre di sviluppare delle funzioni per la verifica dell'appartenenza di un numero all'insieme dei numeri primi e per l'acquisizione e validazione stretta degli input, dal momento che queste operazioni vengono ripetute numerose volte all'interno dell'algoritmo.

0.3.2 Passi dell'algoritmo

I passi dell'algoritmo per risolvere il problema sono i seguenti:

- Acquisire l'operazione scelta dall'utente.
- In base alla scelta effettuata:
  - Acquisire le tre basi e i tre esponenti;
  - Verificare se la relazione $a^x + b^y = c^z$ è soddisfatta dai valori inseriti;
  - Verificare se le tre basi hanno fattori primi in comune;
  - Stampare a schermo gli eventuali fattori primi individuati.
- oppure:
  - Acquisire un numero intero maggiore di zero;
  - Verificare se il numero acquisito è pari o dispari;
  - Applicare la funzione al valore inserito;
  - Ripetere l'applicazione ai valori mano a mano ottenuti, finchè diversi da 1;
  - Stampare la sequenza di valori ottenuti.
- oppure:
  - Acquisire e validare i due numeri primi consecutivi;
  - Calcolare il valore assoluto della differenza dei due numeri;
  - Calcolare la potenza quadrata del logaritmo naturale del numero minore;
  - Verificare se il valore assoluto della differenza è minore di $(ln(min(p_1, p_2)))^2$;
  - Stampare i due valori ottenuti;
- Stampare l'esito della verifica.

0.4 Implementazione dell'algoritmo

```
/*********************************************************************/
/* PROGRAMMA PER VERIFICARE LE CONGETTURE DI BEAL, COLLATZ E CRAMER */
/*********************************************************************/

/* INCLUSIONE DELLE LIBRERIE */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* DEFINIZIONE DEI TIPI */

/* Tipo ricorsivo per creare liste dinamiche */
typedef struct lista
{
  unsigned int elemento; /* numero intero */
  struct lista *succ;    /* indirizzo dell'elemento successivo */
} lista_t;

/* DICHIARAZIONE DELLE FUNZIONI */
int acquisisci_operazione(void);
void acquisisci_valori(unsigned int *,
                       char *,
             int);
int primo(unsigned int);
int verifica_beal(lista_t **);
int verifica_collatz(lista_t **);
int verifica_cramer(void);
int inserisci_elemento(lista_t **,
                       unsigned int);
void stampa_lista(lista_t *);

/*****************************/
/* DEFINIZIONE DELLE FUNZONI */
/*****************************/

/* DEFINIZIONE DELLA FUNZIONE MAIN */
int main(void)
{
   /* Dichiarazione variabili locali alla funzione*/
   int op;          /* INPUT: operazione scelta dall'utente */
   lista_t *valori = NULL; /* OUTPUT: i valori prodotti dalla congettura scelta */
   int esito;       /* OUTPUT: l'esito della verifica */

   /* Stampare il titolo del programma */
   printf("\nPROGRAMMA PER LA VERIFICA DI CONGETTURE\n");
   putchar('\n');

   /* Acquisire l'operazione scelta dall'utente */
   op = acquisisci_operazione();

   /* Verificare la congettura scelta dall' utente */
   switch (op)
   {
     case 1:
       esito = verifica_beal(&valori);
       break;
     case 2:
       esito = verifica_collatz(&valori);
```

```
      break;
    case 3:
      esito = verifica_cramer();
      break;
  }

  /* Stampare l'esito complessivo della verifica */
  putchar('\n');
  printf("Congettura %s.\n",
         (esito)?
         "verificata":
         "non verificata");

  return(0);
}

/* DEFINIZIONE DELLA FUNZIONE PER SCEGLIERE LA CONGETTURA DA VERIFICARE */
int acquisisci_operazione(void)
{
  /* Dichiarazione delle variabili locali alla funzione */
  int operazione;          /* INPUT/OUTPUT: operazione scelta dall'utente */
  int esito_acquisizione,  /* LAVORO: esito della scanf */
      acquisizione_errata; /* LAVORO: esito globale dell'acquisizione */

  do
  {
    /* Stampare le istruzioni per la selezione */
    putchar('\n');
    printf("Scegliere una congettura da verificare:\n\n");
    printf("(1) - Beal\n");
    printf("(2) - Collatz\n");
    printf("(3) - Cramer.\n");

    printf("Digitare la propria scelta e premere invio: ");

    /* Acquisire l'operazione scelta */
    esito_acquisizione = scanf("%d",
                               &operazione);

    /* Validare l'acquisizione */
    acquisizione_errata = esito_acquisizione != 1 || operazione < 1 || operazione > 3;

    /* Stampare un messaggio di errore in caso di acquisizione errata */
    if (acquisizione_errata)
      printf("Valore non accettabile!\n");
    while (getchar() != '\n');
  }
  while (acquisizione_errata);

  return(operazione);
}

/* DEFINIZIONE DELLA FUNZIONE PER VERIFICARE SE UN NUMERO  PRIMO */
int primo(unsigned int valore) /* INPUT: il valore da controllare */
{
  /* Dichiarazione delle variabili locali */
  int primo;      /* OUTPUT: esito della verifica */
  unsigned int i; /* LAVORO: indice di scorrimento */

  if (valore == 2)
    primo = 1;
```

```
   /* Verificare se il valore ha divisori */
   for (i = 2,
          primo = 1;
        i < sqrt(valore) && primo != 0;
        i++)
  {
      if(valore % i == 0)
         primo = 0;
    }

   return(primo);
}

/* Definizione della funzione per acquisire e validare valori numerici */
void acquisisci_valori(unsigned int *valore, /*INPUT/OUTPUT: valore da acquisire */
                       char *messaggio,      /* LAVORO: messaggio specifico */
                       int min)              /* LAVORO: limite inferiore per l'acquisizione */
{
   /* Dichiarazione delle variabili locali */
   int esito_acquisizione,  /* LAVORO: esito della scanf*/
       acquisizione_errata; /* LAVORO: esito complessivo dell' acquisizione */

   do
   {
     /* Stampare il messaggio specifico */
     printf("Digita %s:  ",
            messaggio);

     /* Acquisire il valore */
     esito_acquisizione = scanf("%u",
                                valore);

     /* Verificare l'esito dell'acquisizione */
     if (min >= 11)
       acquisizione_errata = esito_acquisizione != 1 || *valore < min || primo(*valore) != 1;
     else
       acquisizione_errata = esito_acquisizione != 1 || *valore < min;

     /* Stampare un messaggio di errore */
     if (acquisizione_errata)
       printf("Valore non accettabile!\n");
     while (getchar() != '\n');
   }
   while (acquisizione_errata);
}

/* DEFINIZIONE DELLA FUNZIONE PER VERIFICARE LA CONGETTURA DI BEAL */
int verifica_beal(lista_t **fattori_comuni) /* OUTPUT: fattori primi comuni trovati */
{
   /* Dichiarazione delle variabili locali */
   unsigned int basi[3], /* INPUT: basi delle tre potenze */
                esp[3]; /* INPUT: esponenti delle tre potenze */
   unsigned int i;     /* LAVORO: indice di scorrimento */
   int div;          /* LAVORO: divisore per la scomposizione in fattori primi */
   int esito_eq, /* OUTPUT: esito della verifica dell' equazione */
       esito;    /* OUTPUT: esito complessivo della verifica */

   /* Stampare la scelta effettuata */
   printf("\nCONGETTURA DI BEAL.\n");

   /* Acquisire le tre potenze */
   for (i = 0;
```

```
     i < 3;
     i++)
{
  putchar('\n');
  acquisisci_valori(basi + i,
                    "una base (>= 1)",
                    1);

  acquisisci_valori(esp + i,
                    "un esponente (>= 3)",
                    3);
}

/* Verificare l' equazione a^x + b^y = c^z */
esito_eq = ((pow(basi[0], esp[0]) + pow(basi[1], esp[1])) == pow(basi[2], esp[2]));

/* Scomporre la prima base in fattori primi */
div = 2;
while (basi[0] > 1)
{
   if (basi[0] % div == 0)
   {
     basi[0] = basi[0] / div;

     /* Verificare se i fattori primi trovati dividono le altre due basi */
     if (primo(div) == 1 && basi[1] % div == 0 && basi[2] % div == 0)
        inserisci_elemento(fattori_comuni,
                           div);
   }
   else
     div += 1;
}

if (esito_eq == 1)
{
  printf("\nL'equazione a^x + b^y = c^z  soddisfatta dai valori inseriti.\n");

  if (*fattori_comuni != NULL)
  {
    printf("Sono stati trovati i seguenti fattori primi comuni:\n");
    stampa_lista(*fattori_comuni);
    esito = 1;
  }
  else
  {
    printf("Non sono stati trovati fattori primi comuni.\n");
    esito = 0;
  }
}
else
{
 printf("\nL'equazione a^x + b^y = c^z non  soddisfatta dai valori inseriti.\n");
   esito = 0;

   if (*fattori_comuni != NULL)
   {
     printf("Sono comunque stati trovati fattori primi comuni:\n");
     stampa_lista(*fattori_comuni);
   }
}

return(esito);
```

```
}

/* DEFINIZIONE DELLA FUNZIONE PER VERIFICARE LA CONGETTURA DI COLLATZ */
int verifica_collatz(lista_t **valori_generati) /* OUTPUT: valori generati dalla funzione */
{
   /* Dichiarazione delle variabili locali alla funzione */
   unsigned int val_corr; /* INPUT: valore inserito dall'utente */
   int esito_ins;         /* LAVORO: esito dell'inserimento nella lista */
   int verifica;          /* OUTPUT: esito della verifica */

   /* Stampare la scelta effettuata */
   printf("\nCONGETTURA DI COLLATZ.\n");

   /* Acquisire il valore di partenza */
   putchar('\n');
   acquisisci_valori(&val_corr,
                     "un numero intero (> 0)",
             1);

   do
   {
      /* Inserire il valore attuale nella lista */
      esito_ins = inserisci_elemento(valori_generati,
                                     val_corr);

      /* Verificare se il valore  pari */
      if (val_corr % 2 == 0)
      {
        val_corr = val_corr / 2;
      }
      else
      {
        val_corr = 3 * val_corr + 1;
      }
   }
   while (val_corr != 1 && esito_ins == 1);

   /* Stabilire l'esito della congettura */
   verifica = (val_corr == 1)?
              1:
              0;

   /* Stampare i valori generati */
   stampa_lista(*valori_generati);
   putchar('1');

   return(verifica);
}

/* DEFINIZIONE DELLA FUNZIONE PER INSERIRE UN INTERO IN UNA LISTA */
int inserisci_elemento(lista_t **pos, /* INPUT/OUTPUT: indirizzo prima locazione */
                       unsigned int valore) /* INPUT: il valore da inserire */
{
   /* Dichiarazione delle variabili locali */
   lista_t *pos_corr,  /* LAVORO: primo puntatore di scorrimento */
           *pos_prec,  /* LAVORO: secondo puntatore di scorrimento */
         *nuova_pos; /* LAVORO: puntatore nuova locazione */
   int esito_ins;     /* OUTPUT: esito dell'inserimento*/

   /* Scorrere la lista fino a trovare una locazione vuota */
   for (pos_corr = pos_prec = *pos;
        pos_corr != NULL && pos_corr->elemento != valore;
```

```
        pos_prec = pos_corr,
          pos_corr = pos_corr->succ);
    if (pos_corr != NULL && pos_corr->elemento == valore)
      esito_ins = 0;
    else
    {
      esito_ins = 1;

      /* Allocare e inizializzare il nuovo elemento */
      nuova_pos = (lista_t *)malloc(sizeof(lista_t));
      nuova_pos->elemento = valore;
      nuova_pos->succ = pos_corr;

      /* Posizionare il nuovo elemento nella lista */
      if (pos_corr == *pos)
        *pos = nuova_pos;
      else
        pos_prec->succ = nuova_pos;
    }

    return(esito_ins);
}

/* DEFINIZIONE DELLA FUNZIONE PER VERIFICARE LA CONGETTURA DI CRAMER */
int verifica_cramer(void)
{
    /* Dichiarazione delle variabili locali */
    unsigned int primo1, /* INPUT: primo numero primo*/
                 primo2; /* INPUT: secondo numero primo*/
    int i,         /* LAVORO: indice di scorrimento */
        acquisizione_errata;  /* LAVORO: esito acquisizione */
    unsigned int abs_diff,  /* OUTPUT: valore assoluto della differenza dei due numeri */
                 pow_ln;    /* OUTPUT: quadrato del ln del numero minore */
    int verifica;         /* OUTPUT: esito della verifica */

    /* Stampare la scelta effettuata */
    printf("\nCONGETTURA DI CRAMER.\n");
    putchar('\n');

    do
    { /* Acquisire il primo numero primo */
      acquisisci_valori(&primo1,
                        "il primo numero primo (> 10)",
                        11);

      /* Acquisire il secondo numero primo */
      acquisisci_valori(&primo2,
                        "il secondo numero primo (maggiore del precedente)",
                        primo1);


      /* Verificare che siano consecutivi */
      for(i = (primo1 + 1),
            acquisizione_errata = 0;
          i < primo2;
        i++)
      {
        if (primo(i) == 1)
          acquisizione_errata = 1;
      }

      /* Stampare messaggio di errore */
```

```
     if (acquisizione_errata)
       printf("\nI due numeri devono essere consecutivi!\n");
   }
   while (acquisizione_errata);

   /* Calcolare il valore assoluto della differenza */
   abs_diff = abs(primo1 -
                  primo2);

   /* Calcolare il quadrato del logaritmo naturale */
   pow_ln = pow(log(primo1),
                2);

   /* Stabilire l'esito della verifica */
   verifica = abs_diff < pow_ln;

   /* Stampare i risultati */
   printf("Il valore assoluto della differenza tra %u e %u  %u.\n",
          primo1,
          primo2,
          abs_diff);

   printf("Il quadrato del logaritmo naturale del numero minore vale %u.\n",
          pow_ln);

   printf("Il valore assoluto della differenza %s minore del quadrato del logaritmo naturale del numero
    minore.\n",
          (verifica)?
          " ":
          "non ");

   return(verifica);
}

/* DEFINIZIONE DELLA FUNZIONE PER STAMPARE UNA LISTA DI NUMERI */
void stampa_lista(lista_t *lista) /* INPUT: l'indirizzo della prima locazione della lista */
{
   lista_t *corr; /* LAVORO: puntatore di scorrimento della lista */

   for (corr = lista;
        corr != NULL;
        corr = corr->succ)
   {
     printf("%u\n",
            corr->elemento);
   }
}
```

0.5 Testing del programma

I test effettuati rivelano che se l'utente sceglie una operazione diversa da 1, 2 o 3, il programma emette un messaggio di errore e chiede di rieffetturare la scelta.
Allo stesso modo, il programma acquisisce correttamente i valori in ingresso in base alle limitazioni imposte dall'operazione scelta dall'utente.
Il programma verifica correttamente se un numero è primo oppure no; ed esegue correttamente la scomposizione in fattori primi.

Test 1
Operazione scelta: 1
Prima potenza: 5^3
Seconda potenza: 7^9
Terza potenza: 8^12
Equazione: non soddisfatta
Fattori primi comuni: nessuno
Esito verifica: non verificata

Test 2
Operazione scelta: 1
Prima potenza: 3^3
Seconda potenza: 6^3
Terza potenza: 3^5
Equazione: soddisfatta
Fattori primi comuni: 3
Esito verifica: verificata

Test 3
Operazione scelta: 1
Prima potenza: 5^3
Seconda potenza: 50^10
Terza potenza: 5^15
Equazione: non soddisfatta
Fattori primi comuni: 5
Esito verifica: non verificata

Test 4
Operazione scelta: 1
Prima potenza: 729^3
Seconda potenza: 1458^3
Terza potenza: 243^4
Equazione: soddisfatta
Fattori primi comuni: 3
Esito verifica: verificata

Test 5
Operazione scelta: 1
Prima potenza: 42^5
Seconda potenza: 24^10
Terza potenza: 60^5
Equazione: non soddisfatta
Fattori primi comuni: 2, 3
Esito verifica: non verificata

Test 6

Operazione scelta: 1
Prima potenza: 42^3
Seconda potenza: 210^3
Terza potenza: 42^5
Equazione: non soddisfatta
Fattori primi comuni: 2, 3, 7
Esito verifica: non verificata

Test 7
Operazione scelta: 1
Prima potenza: 7^62
Seconda potenza: 7^7
Terza potenza: 98^3
Equazione: soddisfatta
Fattori primi comuni: 7
Esito verifica: verificata

Test 8
Operazione scelta: 1
Prima potenza: 1000^3
Seconda potenza: 2500^{10}
Terza potenza: 8591^6
Equazione: non soddisfatta
Fattori primi comuni: nessuno
Esito verifica: non verificata

Test 9
Operazione scelta: 1
Prima potenza: 9^3
Seconda potenza: 27^3
Terza potenza: 81^3
Equazione: non soddisfatta
Fattori primi comuni: 3
Esito verifica: non verificata

Test 10
Operazione scelta: 1
Prima potenza: 550^5
Seconda potenza: 825^5
Terza potenza: 275^6
Equazione: soddisfatta
Fattori primi comuni: 5, 11
Esito verifica: verificata

Test 11
Operazione scelta: 2
Numero di partenza: 5
Valori ottenuti: 16, 8, 4, 2, 1
Esito verifica: verificata

Test 12
Operazione scelta: 2

Numero di partenza: 13
Valori ottenuti: 40, 20, 10, 5, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 13
Operazione scelta: 2
Numero di partenza: 42
Valori ottenuti: 21, 64, 32, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 14
Operazione scelta: 2
Numero di partenza: 81
Valori ottenuti: 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 15
Operazione scelta: 2
Numero di partenza: 47
Valori ottenuti: 142, 71, 214, 107, 322, 161, 484, 242, 121, 364, 182, 91, 274, 137, 412, 206, 103, 310, 155, 466, 233, 700, 350, 175, 526, 263, 790, 395, 1186, 593, 1780, 890, 445, 1336, 668, 334, 167, 502, 251, 754, 377, 1132, 566, 283, 850, 425, 1276, 638, 319, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 16
Operazione scelta: 2
Numero di partenza: 120
Valori ottenuti: 60, 30, 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2
Esito verifica: verificata

Test 17
Operazione scelta: 2
Numero di partenza: 567
Valori ottenuti: 1702, 851, 2554, 1277, 3832, 1916, 958, 479, 1438, 719, 2158, 1079, 3238, 1619, 4858, 2429, 7288, 3644, 1822, 911, 2734, 1367, 4102, 2051, 6154, 3077, 9232, 4616, 2308, 1154, 577, 1732, 866, 433, 1300, 650, 325, 976, 488, 244, 122, 61, 184, 92, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 18
Operazione scelta: 2
Numero di partenza: 2747
Valori ottenuti: 8242, 4121, 12364, 6182, 3091, 9274, 4637, 13912, 6956, 3478, 1739, 5218, 2609, 7828, 3914, 1957, 5872, 2936, 1468, 734, 367, 1102, 551, 1654, 827, 2482, 1241, 3724, 1862, 931, 2794, 1397, 4192, 2096, 1048, 524, 262, 131, 394, 197, 592, 296, 148, 74, 37, 112, 56, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 19
Operazione scelta: 2
Numero di partenza: 1200
Valori ottenuti: 600, 300, 150, 75, 226, 113, 340, 170, 85, 256, 128, 64, 32, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 20
Operazione scelta: 2
Numero di partenza: 30
Valori ottenuti: 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
Esito verifica: verificata

Test 21
Operazione scelta: 3
Primo numero primo: 11
Secondo numero primo: 13
Valore assoluto della differenza: 2
Quadrato del logaritmo naturale del numero minore: 5
Esito verifica: verificata

Test 22
Operazione scelta: 3
Primo numero primo: 53
Secondo numero primo: 59
Valore assoluto della differenza: 6
Quadrato del logaritmo naturale del numero minore: 15
Esito verifica: verificata

Test 23
Operazione scelta: 3
Primo numero primo: 103
Secondo numero primo: 107
Valore assoluto della differenza: 4
Quadrato del logaritmo naturale del numero minore: 21
Esito verifica: verificata

Test 24
Operazione scelta: 3
Primo numero primo: 257
Secondo numero primo: 263
Valore assoluto della differenza: 6
Quadrato del logaritmo naturale del numero minore: 30
Esito verifica: verificata

Test 25
Operazione scelta: 3
Primo numero primo: 401
Secondo numero primo: 409
Valore assoluto della differenza: 8
Quadrato del logaritmo naturale del numero minore: 35
Esito verifica: verificata

Test 26
Operazione scelta: 3
Primo numero primo: 1279

Secondo numero primo: 1283
Valore assoluto della differenza: 4
Quadrato del logaritmo naturale del numero minore: 51
Esito verifica: verificata

Test 27
Operazione scelta: 3
Primo numero primo: 3163
Secondo numero primo: 3167
Valore assoluto della differenza: 4
Quadrato del logaritmo naturale del numero minore: 64
Esito verifica: verificata

Test 28
Operazione scelta: 3
Primo numero primo: 1879
Secondo numero primo: 1889
Valore assoluto della differenza: 10
Quadrato del logaritmo naturale del numero minore: 56
Esito verifica: verificata

Test 29
Operazione scelta: 3
Primo numero primo: 4931
Secondo numero primo: 4933
Valore assoluto della differenza: 2
Quadrato del logaritmo naturale del numero minore: 72
Esito verifica: verificata

Test 30
Operazione scelta: 3
Primo numero primo: 19
Secondo numero primo: 23
Valore assoluto della differenza: 4
Quadrato del logaritmo naturale del numero minore: 8
Esito verifica: verificata

0.6 Verifica del Programma

0.6.1 Brano di Codice Scelto

```
if (val_corr % 2 == 0)
{
  val_corr = val_corr / 2;
}
else
{
  val_corr = 3 * val_corr + 1;
}
```

0.6.2 Proprietà da Verificare

Se denotiamo con:

- $val(\texttt{val\_corr})$ il valore contenuto in `val_corr` all'inizio dell'esecuzione;
- $f(val(\texttt{val\_corr}))$ il valore contenuto in `val_corr` al termine dell'esecuzione;

e definita la funzione $f(val(\texttt{val\_corr}))$ come:

$$f(val(\texttt{val\_corr})) = \begin{cases} val(\texttt{val\_corr})/2, & \text{se } val(\texttt{val\_corr}) \text{ pari} \\ 3 \cdot val(\texttt{val\_corr}) + 1, & \text{se } val(\texttt{val\_corr}) \text{ dispari} \end{cases}$$

la proprietà da verificare è che:

$$\texttt{val\_corr} = f(val(\texttt{val\_corr}))$$

0.6.3 Svolgimento

E' possibile verificare la correttezza di istruzioni di selezione tramite regole di Dijkstra. La precondizione più debole affinchè la tripla

$$\{Q\}S_1S_2\{R\}$$

sia soddisfatta è definita come

$wp(S,R) = R_{\texttt{val\_corr},f(val(\texttt{val\_corr}))} = (\texttt{val\_corr} = f(val(\texttt{val\_corr})))_{\texttt{val\_corr},f(val(\texttt{val\_corr}))}$

ed è definibile nel modo seguente:

$wp(S,R) = (pari \rightarrow wp(S_1,R)) \wedge (\neg pari \rightarrow wp(S_2,R))$

$wp(S_1,R) = (val(\texttt{val\_corr})\%2 = 0 \rightarrow (\texttt{val\_corr} = f(val(\texttt{val\_corr})))_{\texttt{val\_corr},val(\texttt{val\_corr})/2}) = (val(\texttt{val\_corr})\%2 = 0 \rightarrow val(\texttt{val\_corr})/2 = f(val(\texttt{val\_corr}))) = vero$

$wp(S_2,R) = (val(\texttt{val\_corr})\%2 \neq 0 \rightarrow (\texttt{val\_corr} = f(val(\texttt{val\_corr})))_{\texttt{val\_corr},3*val(\texttt{val\_corr})+1}) = (val(\texttt{val\_corr})\%2 \neq 0 \rightarrow 3 * val(\texttt{val\_corr}) + 1 = f(val(\texttt{val\_corr}))) = vero$

$wp(S,R) = (vero \wedge vero) = vero$

Il brano di codice scelto produce dunque il risultato corretto in ogni caso, a prescindere dallo stato iniziale della computazione.