Exercícios – Compiladores

1) Traduzir os códigos dos programas a seguir para a representação de três endereços.

2) A partir do exercício 1, construir os blocos básicos e CFGs para cada um dos programas.

3) Aplicar uma otimização de código (utilize a que foi determinada para cada grupo) sobre cada um dos programas a seguir. Nos casos onde a otimização não é aplicável sobre o código dos programas, justifique por que a otimização não pode ser aplicada e informe se alguma outra otimização (dentre as aprendidas na disciplina) poderia ser aplicada.

OBSERVAÇÃO: Na resolução de três endereços dos programas a seguir, você deve:

- ignorar os comentários
- considerar que todas as variáveis foram declaradas
- considerar que todas as funções e/ou chamadas a procedimentos são válidas
- não é necessário gerar código de três endereços para declarações de variáveis

```
================================================================================
int main()
{
  int teste=1, n;
  while (scanf("%d", &n) == 1 && n != -1)
    printf("Teste %d\n%d\n\n", teste++, ((1 << n)+1)*((1 << n)+1));
  return 0;
}
```

```
int main()
{
  int max, objetivo, soma, teste=1, i, j;
  while (scanf("%d %d %d", &x, &y, &n) == 3 && x > 0)
    {
      soma = 0;
      for (i = 0; i < n; i++)
        {
          scanf("%d", &valores[i]);
          soma += valores[i];
        }
      printf("Teste %d\n", teste++);
      if ((soma + x + y) % 2 != 0)
        printf("N\n\n");
      else
        {
          objetivo = (soma + x + y) / 2 - x;
          memset(possivel, 0, sizeof(possivel));
          possivel[0] = 1;
          max = 0;
          for (i = 0; i < n && !possivel[objetivo]; i++)
            {
              for (j = max; j >= 0; j--)
                if (possivel[j])
                  possivel[j+valores[i]] = 1;
              max += valores[i];
            }
          printf("%c\n\n", possivel[objetivo]? 'S' : 'N');
        }
    }
  return 0;
}
```

```
int main()

{
  int i, indice_melhor, n;
  int turma=1;
  struct
  {
    int codigo, media;
  } alunos[MAX_ALUNOS];

  scanf("%d", &n);
  while (n > 0)
    {
      /* le dados dos alunos */
      for (i = 0; i < n; i++)
```

```
        scanf("%d %d", &alunos[i].codigo, &alunos[i].media);
      /* procura aluno de maior media */
      indice_melhor = 0;
      for (i = 1; i < n; i++)
        if (alunos[i].media > alunos[indice_melhor].media)
            indice_melhor = i;
      /* escreve resposta */
      printf("Turma %d\n", turma++);
      for (i = 0; i < n; i++)
        if (alunos[i].media == alunos[indice_melhor].media)
          printf("%d ", alunos[i].codigo);
      printf("\n\n");
      scanf("%d", &n);
    }
  return 0;
}
```

```
int main()

{
  int n, total_joao, total_ze, deposito_joao, deposito_ze, teste=1, i;
  while (scanf("%d", &n) == 1 && n > 0)
    {
      total_joao = 0;
      total_ze = 0;
      printf("Teste %d\n", teste++);
      for (i = 0; i < n; i++)
        {
          scanf("%d %d", &deposito_joao, &deposito_ze);
          total_joao += deposito_joao;
          total_ze += deposito_ze;
          printf("%d\n", total_joao - total_ze);
        }
      printf("\n");
    }
  return 0;
}
```

```
int main()

{
   int teste=0; /* Teste atual */
   int participantes, rodadas; /* Numero de participantes e de rodadas */
   int part[PMAX]; /* Participante atualmente na posicao */
   int npart; /* Numero de participantes da rodada */
   int ordem; /* Ordem dada pelo chefe */
   int acao; /* Acao atual */
   int posacao; /* Posicao de quem executa a acao atualmente */
   int i,j,k; /* Iteradores */

   scanf("%d %d", &participantes, &rodadas);
   while(participantes || rodadas) { /* Termina quando ambos forem 0 */
      printf("Teste %d\n", ++teste);

      for(i=0;i<participantes;i++) scanf("%d",&(part[i])); /* Le posicao dos
                                                              participantes */

      for(i=0;i<rodadas;i++)
      {
         scanf("%d %d", &npart, &ordem);
```

```
         posacao=0; /* A primeira acao sera executada pelo primeiro da fila
                       sempre */
         for(j=0;j<npart;j++)
         {
            scanf("%d",&acao);

            /* Participante acertou - Passa para o proximo */
            if(acao == ordem) posacao++;

            /* Participante errou - Elimina o participante */
            else for(k=posacao+1;k<npart;k++) part[k-1]=part[k];
         }
      }

      printf("%d\n\n", part[0]);

      scanf("%d %d", &participantes, &rodadas);
   }

   return 0;
}
```

==========

```
int main()
{
   int teste=0; /* Teste atual */
   int comprimento; /* Comprimento do cartao */
   int nfrases; /* Numero de frases */
   int nchar[MAXFRASES+1]; /* Numero de caracteres da frase */
   int ndesculpe[MAXFRASES+1]; /* Vezes que ocorre "desculpe" */
   int res[MAXFRASES+1][MAXCARTAO+1]; /* Resultado para i frases num
                                         cartao j */
   int i,j; /* Iteradores */


   scanf("%d %d", &comprimento, &nfrases);
   while(comprimento || nfrases) { /* Termina quando ambas forem 0 */
      printf("Teste %d\n", ++teste);

      for(i=1;i<=nfrases;i++) /* Le a entrada */
         scanf("%d %d", &(nchar[i]), &(ndesculpe[i]));

   /* Problema da mochila 0-1:
   A. c[i][j] = 0                                          , se i=0 ou j=0
   B. c[i][j] = c[i-1][j]                                  , se peso[i] > j
   C. c[i][j] = max(valor[i]+c[i-1][j-peso[i]],c[i-1][j]), se i>0 e j > peso[i]
   */

      for(i=0;i<=comprimento;i++)
         res[0][i] = 0; /* A. */
      for(i=1;i<=nfrases;i++)
      {
         res[i][0] = 0; /* A. */
         for(j=1;j<=comprimento;j++)
         {
            if(nchar[i] > j)
            {
               res[i][j] = res[i-1][j]; /* B. */
            }
            else /* C. */
            {
```

```
                if((ndesculpe[i]+res[i-1][j-nchar[i]]) > res[i-1][j])
                   res[i][j] = ndesculpe[i]+res[i-1][j-nchar[i]];
                else res[i][j] = res[i-1][j];
             }
          }
       }

       printf("%d\n\n", res[nfrases][comprimento]);

       scanf("%d %d", &comprimento, &nfrases);
    }

    return 0;
}
```